Stephen L. Chiu
Rockwell Science Center
Thousands Oaks, California 91360

Fuzzy Model Identification Based on Cluster Estimation

Abstract

We present an efficient method for estimating cluster centers of numerical data. This method can be used to determine the number of clusters and their initial values for initializing iterative optimization-based clustering algorithms such as fuzzy C-means. Here we use the cluster estimation method as the basis of a fast and robust algorithm for identifying fuzzy models. A benchmark problem involving the prediction of a chaotic time series shows this model identification method compares favorably with other, more computationally intensive methods. We also illustrate an application of this method in modeling the relationship between automobile trips and demographic factors. © 1994 John Wiley and Sons, Inc.

Introduction

Clustering of numerical data forms the basis of many classification and system modeling algorithms. The purpose of clustering is to distill natural groupings of data from a large data set, producing a concise representation of a system's behavior. In particular, the fuzzy C-means (FCM) clustering algorithm (Dunn, 1974; Bezdek, 1974; Bezdek et al., 1987) has been widely studied and applied. The FCM algorithm is an iterative optimization algorithm that minimizes the cost function

$$J = \sum_{k=1}^{n} \sum_{i=1}^{c} \mu_{ik}^{m} \|x_k - v_i\|^2$$

where n is the number of data points, c is the number of clusters, x_k is the kth data point, v_i is the ith cluster center, μ_{ik} is the degree of membership of the kth data in the ith cluster, and m is a constant greater than 1 (typically $m = 2$). The degree of membership μ_{ik} is defined by

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|x_k - v_i\|}{\|x_k - v_j\|} \right)^{2/(m-1)}} . \qquad (1)$$

Starting with a desired number of clusters c and an initial guess for each cluster center v_i, $i = 1, 2, \ldots, c$, FCM will converge to a solution for v_i that represents either a local minimum or a saddle point of the cost function (Bezdek et al., 1987). The quality of the FCM solution, like that of most nonlinear optimization problems, depends strongly on the choice of initial values (i.e., the number c and the initial cluster centers).

Yager and Filev (1992) proposed a simple and effective algorithm, called the Mountain Method, for estimating the number and initial locations of cluster centers. Their method is based on making a grid of the data space and computing a potential value for each grid point based on its distances to the actual data points; a grid point with many data points nearby will have a high potential value. The grid point with the highest potential value is chosen as the first cluster center. The key idea in their method is that once the first cluster center is chosen, the potential of all grid points are reduced according to their distance from the cluster center. Grid points near the first cluster center will have greatly reduced potential. The next cluster center is then placed at the grid point with the highest remaining potential value. This procedure of acquiring new cluster center and reducing the potential of surrounding grid points repeats until the potential of all grid points falls below a threshold. Although this method is simple and effective, the computation grows exponentially with the dimension of the problem. For example, a cluster-

Journal of Intelligent and Fuzzy Systems, Vol. 2, 267–278 (1994)
© 1994 John Wiley & Sons, Inc.
CCC 1064-1246/94/030267-12

ing problem with 4 variables and each dimension having a resolution of 10 grid lines would result in 10^4 grid points that must be evaluated.

We present a modified form of the Mountain Method for cluster estimation. We consider each *data point*, not a grid point, as a potential cluster center. Using this method, the number of effective "grid points" to be evaluated is simply equal to the number of data points, independent of the dimension of the problem. Another advantage of this method is that it eliminates the need to specify a grid resolution, in which trade-offs between accuracy and computational complexity must be considered. We also improve the computational efficiency and robustness of the original method via other modifications.

Although clustering is generally associated with classification problems, here we use the cluster estimation method as the basis of a fuzzy model identification algorithm. The key to the speed of this model identification algorithm is that it does not involve any iterative nonlinear optimization; in addition, the computation grows only linearly with the dimension of the problem. We use a benchmark problem in chaotic time series prediction to compare the performance of this algorithm with the published results of other algorithms. We also show an application: estimating the number of automobile trips generated from an area based on its demographics.

Cluster Estimation

Consider a collection of n data points $\{x_1, x_2, \ldots, x_n\}$ in an M-dimensional space. Without loss of generality, we assume that the data points have been normalized in each dimension so that their coordinate ranges in each dimension are equal, i.e., the data points are bounded by a hypercube. We consider each data point as a potential cluster center and define a measure of the potential of data point x_i as

$$P_i = \sum_{j=1}^{n} \mathbf{e}^{-\alpha \|x_i - x_j\|^2}$$

where

$$\alpha = \frac{4}{r_a^2} \tag{2}$$

and r_a is a positive constant. Thus, the measure of potential for a data point is a function of its distances to all other data points. A data point with many neighboring data points will have a high potential value. The constant r_a is effectively the radius defining a neighborhood; data points outside this radius have little influence on the potential. This measure of potential differs from that proposed by Yager and Filev in two ways: (1) the potential is associated with an actual data point instead of a grid point; (2) the influence of a neighboring data point decays exponentially with the square of the distance instead of the distance itself. Using the square of the distance eliminates the square root operation that otherwise would be needed for determining the distance itself.

After the potential of every data point has been computed, we select the data point with the highest potential as the first cluster center. Let x_1^* be the location of the first cluster center and P_1^* be its potential value. We then revise the potential of each data point x_i by the formula

$$P_i \Leftarrow P_i - P_1^* \mathbf{e}^{-\beta \|x_i - x_1^*\|^2} \tag{3}$$

where

$$\beta = \frac{4}{r_b^2}$$

and r_b is a positive constant. Thus, we subtract an amount of potential from each data point as a function of its distance from the first cluster center. The data points near the first cluster center will have greatly reduced potential, and therefore are unlikely to be selected as the next cluster center. The constant r_b is effectively the radius defining the neighborhood that will have measurable reductions in potential. To avoid obtaining closely spaced cluster centers, we set r_b to be somewhat greater than r_a; a good choice is $r_b = 1.5 r_a$.

When the potential of all data points have been revised according to eq. (3), we select the data point with the highest remaining potential as the second cluster center. We then further reduce the potential of each data point according to their distance to the second cluster center. In general, after the kth cluster center has been obtained, we revise the potential of each data point by the formula

$$P_i \Leftarrow P_i - P_k^* \mathbf{e}^{-\beta \|x_i - x_k^*\|^2}$$

where x_k^* is the location of the kth cluster center and P_k^* is its potential value.

In Yager and Filev's procedure, the process of acquiring new cluster center and revising potentials repeats until

$$P_k^* < \varepsilon P_1^*$$

where ε is a small fraction. The choice of ε is an important factor affecting the results; if ε is too large, too few data points will be accepted as cluster centers; if ε is too small, too many cluster centers will be generated. We have found it difficult to establish a single value for ε that works well for all data patterns, and have therefore developed additional criteria for accepting/rejecting cluster centers. We use the following criteria:

```
if $P_k^* > \bar{\varepsilon} P_1^*$
  Accept $x_k^*$ as a cluster center and continue.
else if $P_k^* < \underline{\varepsilon} P_1^*$
  Reject $x_k^*$ and end the clustering process.
else
  Let $d_{min}$ = shortest of the distances between
    $x_k^*$ and all previously found cluster
    centers.
  if $\frac{d_{min}}{r_a} + \frac{P_k^*}{P_1^*} \geq 1$
    Accept $x_k^*$ as a cluster center and continue.
  else
    Reject $x_k^*$ and set the potential at $x_k^*$ to 0.
    Select the data point with the next highest
    potential as the new $x_k^*$ and re-test.
  end if
end if
```

Here $\bar{\varepsilon}$ specifies a threshold for the potential above which we will definitely accept the data point as a cluster center; $\underline{\varepsilon}$ specifies a threshold below which we will definitely reject the data point. We use $\bar{\varepsilon} = 0.5$ and $\underline{\varepsilon} = 0.15$. If the potential falls in the gray region, we check if the data point provides a good trade-off between having a sufficient potential and being sufficiently far from existing clusters centers.

Model Identification

When we apply the cluster estimation method to a collection of input/output data, each cluster center is in essence a prototypical data point that exemplifies a characteristic behavior of the system. Hence, each cluster center can be used as the basis of a rule that describes the system behavior.

Consider a set of c cluster centers $\{x_1^*, x_2^*, \ldots, x_c^*\}$ in an M-dimensional space. Let the first N dimensions correspond to input variables and the last M–N dimensions correspond to output variables. We decompose each vector x_i^* into two component vectors y_i^* and z_i^*, where y_i^* contains the first N elements of x_i^* (i.e., the coordinates of the cluster center in input space) and z_i^* contains the last M–N elements (i.e., the coordinates of the cluster center in output space).

We consider each cluster center x_i^* as a fuzzy rule that describes the system behavior. Given an input vector y, the degree to which rule i is fulfilled is defined as

$$\mu_i = \mathbf{e}^{-\alpha \|y \;\; y_i^*\|^2} \tag{4}$$

where α is the constant defined by eq. (2). We compute the output vector z via

$$z = \frac{\sum_{i-1}^{c} \mu_i z_i^*}{\sum_{i-1}^{c} \mu_i} . \tag{5}$$

We can view this computational model in terms of a fuzzy inference system employing traditional fuzzy if-then rules. Each rule has the following form:

if Y_1 is A_1 & Y_2 is A_2 &. . . then Z_1
is B_1 & Z_2 is B_2. . .

where Y_j is the jth input variable and Z_j is the jth output variable; A_j is an exponential membership function and B_j is a singleton. For the ith rule that is represented by cluster center x_i^*, A_j and B_j are given by

$$A_j(q) = \mathbf{e}^{-\alpha(q - y_{ij}^*)^2} \tag{6}$$

$$B_j = z_{ij}^* \tag{7}$$

where y_{ij}^* is the jth element of y_i^* and z_{ij}^* is the jth element of z_i^*. Our computational scheme is equivalent to an inference method that uses multiplication as the AND operator, weights the output of each rule by the rule's firing strength, and computes the output value as a weighted average of the output of each rule. It is also interesting to note the correspondence between eqs. (4) and (5) and the Radial Basis Functions (RBF) approach to modeling (Powell, 1987), which suggests an alternative interpretation of this model as a form of RBF model.

Eqs. (4) and (5) provide a simple and direct way to translate a set of cluster centers into a fuzzy model. Yager and Filev (1993) have similarly used the cluster centers extracted by the Mountain Method to form a fuzzy model; to optimize the rules, they used backpropagation to iteratively adjust the y_{ij}^*, z_{ij}^*, and individual α_{ij} parameters in eqs. (6) and (7). We take a different approach by allowing z_i^* in eq. (5) to be a linear function of the input variables, instead of a simple constant. That is, we let

$$z_i^* = G_i y + h_i \tag{8}$$

where G_i is an $(M-N) \times N$ constant matrix, and h_i is a constant column vector with M–N elements. The equivalent if-then rules then become the Takagi–Sugeno type (Takagi and Sugeno, 1985), where the consequent of each rule is a linear equation in the input variables. Models that employ the Takagi–Sugeno type rules have been shown to be able to accurately represent complex behavior with only a few rules (Sugeno and Tanaka, 1991). We will refer to fuzzy models that set z_i^* as a constant as "0th order models," and those that employ eq. 8 as "1st order models."

Expressing z_i^* as a linear function of the input allows a significant degree of rule optimization to be performed without adding much computational complexity. As pointed out by Takagi and Sugeno (1985), given a set of rules with fixed premises, optimizing the parameters in the consequent equations with respect to training data reduces to a linear least-squares estimation problem. Such problems can be solved easily and the solution is always globally optimal.

To convert the equation parameter optimization problem into the linear least-squares estimation problem, we define

$$\rho_i = \frac{\mu_i}{\sum_{j=1}^{c} \mu_j} .$$

Eq. (5) can then be rewritten as

$$z = \sum_{i-1}^{c} \rho_i z_i^* = \sum_{i=1}^{c} \rho_i (G_i y + h_i)$$

or

$$z^{\mathrm{T}} = [\rho_1 y^{\mathrm{T}} \quad \rho_1 \cdots \rho_c y^{\mathrm{T}} \quad \rho_c] \begin{bmatrix} G_1^{\mathrm{T}} \\ h_1^{\mathrm{T}} \\ \vdots \\ G_c^{\mathrm{T}} \\ h_c^{\mathrm{T}} \end{bmatrix}$$

where z^{T} and y^{T} are row vectors. Given a collection of n input data points $\{y_1, y_2, \ldots, y_n\}$, the resultant collection of model output is given by

$$\begin{bmatrix} z_1^{\mathrm{T}} \\ \vdots \\ z_n^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \rho_{1,1} y_1^{\mathrm{T}} & \rho_{1,1} & \cdots & \rho_{c,1} y_1^{\mathrm{T}} & \rho_{c,1} \\ & & \vdots & & \\ \rho_{1,n} y_n^{\mathrm{T}} & \rho_{1,n} & \cdots & \rho_{c,n} y_n^{\mathrm{T}} & \rho_{c,n} \end{bmatrix} \begin{bmatrix} G_1^{\mathrm{T}} \\ h_1^{\mathrm{T}} \\ \vdots \\ G_c^{\mathrm{T}} \\ h_c^{\mathrm{T}} \end{bmatrix} \tag{9}$$

where $\rho_{i,j}$ denotes ρ_i evaluated at y_j. Note that given $\{y_1, y_2, \ldots, y_n\}$, the first matrix on the right hand side of eq. 9 is constant, while the second matrix contains all the parameters to be optimized. To minimize the squared error between the model output and that of the training data, we solve the linear least-squares estimation problem given by Eq. (9), replacing the matrix on the left hand side by the actual output of the training data. Of course, implicit in the least-squares estimation problem is the assumption that the number of training data is greater than the number of parameters to be optimized.

Using the standard notation adopted in most literature, the least-squares estimation problem of eq. (9) has the form

$$AX = B$$

where B is a matrix of output values, A is a constant matrix, and X is a matrix of parameters to be estimated. The well-known pseudo-inverse solution that minimizes $\|AX - B\|^2$ is given by

$$X = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}B .$$

However, computing $(A^{\mathrm{T}}A)^{-1}$ is computationally expensive when $(A^{\mathrm{T}}A)$ is a large matrix ($A^{\mathrm{T}}A$ is $c(N+1) \times c(N+1)$); numerical problems also arise when $(A^{\mathrm{T}}A)$ is nearly singular. We use another well-known method for solving for X, a procedure often referred to as *recursive least-squares estimation* (Astrom and Wittenmark, 1984; Strobach, 1990). This is a computationally efficient and well-behaved method that determines X via the iterative formulae

$$X_{i+1} = X_i + S_{i+1}a_{i+1}(b_{i+1}^{\mathrm{T}} - a_{i+1}^{\mathrm{T}}X_i) \quad (10)$$

$$S_{i+1} = S_i - \frac{S_i a_{i+1} a_{i+1}^{\mathrm{T}} S_i}{1 + a_{i+1}^{\mathrm{T}} S_i a_{i+1}}, \; i = 0, 1, \ldots, n-1 \quad (11)$$

where X_i is the estimate of X at the ith iteration, S_i is a $c(N+1) \times c(N+1)$ covariance matrix, a_i^{T} is the ith row vector of A, and b_i^{T} is the ith row vector of B. The least-squares estimate of X corresponds to the value of X_n. The initial conditions for this iterative procedure are $X_0 = 0$ and $S_0 = \gamma I$, where I is an identity matrix and γ is a large positive value.

To summarize, the model identification method consists of two distinct steps: (1) find cluster centers to establish the number of fuzzy rules and the rule premises, and (2) optimize the rule consequents. Neither of these steps involve nonlinear optimization and both steps have well-bounded computation time. In step 1, the bulk of the computation time is consumed by evaluating the initial potential of each data point. Each subsequent iteration to select a cluster center and subtract potential consumes the same amount of time as evaluating the potential of one data point. Assuming the number of cluster centers that will be obtained is much less than the total number of data points, we can accurately estimate the computation time of step 1 based on the number of data points alone. However, the number of cluster centers found in step 1 affects the computation time of step 2 linearly, because the number of parameters to be optimized grows linearly with the number of clusters. Hence, we can determine the computation time of step 2 only after step 1 is completed.

Although the number of clusters (or rules) is automatically determined by this method, we should note that the user-specified parameter r_a (i.e., the radius of influence of a cluster center) strongly affects the number of clusters that will be generated. A large r_a generally results in fewer clusters and hence a coarser model, while a small r_a can produce excessive number of clusters and a model that does not generalize well (i.e., by over-fitting the training data). Therefore, we may regard r_a as an approximate specification of the desired resolution of the model, which can be adjusted based on the resultant complexity and generalization ability of the model.

RESULTS

We will first apply the model identification method to a simple 2-dimensional function-approximation problem to illustrate some of its properties. Next we will consider a benchmark problem involving the prediction of a chaotic time series and compare the performance of this method with the published results of other methods. Lastly, we will show an application modeling the relationship between the number of automobile trips generated from an area and the demographics of the area.

FUNCTION APPROXIMATION

For illustrative purposes, we consider the simple problem of modeling the nonlinear function

$$z = \frac{\sin(y)}{y}.$$

For the range $[-10, 10]$, we used equally spaced y values to generate 100 training data points. Because the training data are normalized before clustering so that they are bounded by a hypercube, we find it convenient to express r_a as a fraction of the width of the hypercube; in this example we chose $r_a = 0.25$. Applying the cluster estimation method to the training data, 7 cluster centers were found. Figure 1 shows the training data and the locations of the cluster centers.

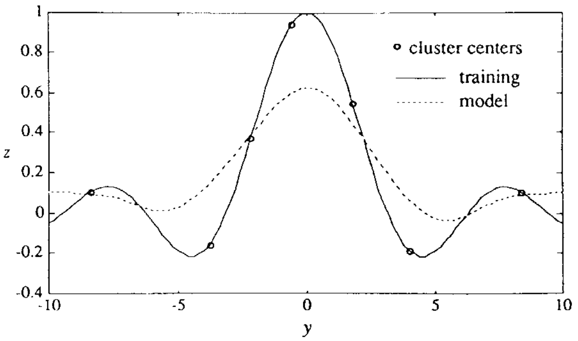

Figure 1. Comparison of training data with unoptimized 0th order model output.

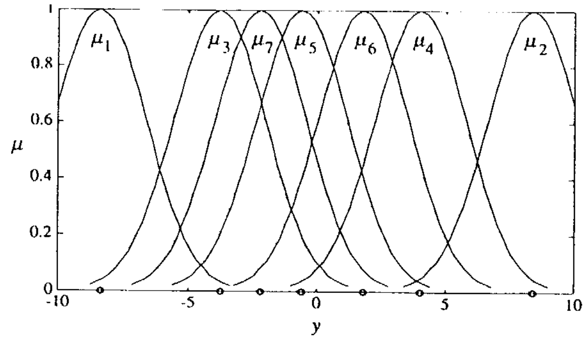

Figure 2. Degree of fulfillment of each rule as a function of input y.

Figure 1 also shows the output of a 0th order fuzzy model that uses constant z_i^* as given by the z coordinate of each cluster center. We see that the modeling error is quite large. Because the clusters are closely spaced with respect to the input dimension, there is significant overlap between the premise conditions of the rules. The degree of fulfillment μ of each rule as a function of y (viz. eq. 4) is shown in Figure 2, where it is evident that several rules can have high firing strength simultaneously even when the input precisely matches the premise condition of one rule. Therefore, the model output is typically a neutral point interpolated from the z coordinates of strongly competing cluster centers.

One way to minimize competition among closely spaced cluster centers is to use the fuzzy C-means definition of μ, viz. eq. (1). When the input precisely matches the location of a cluster center, the FCM definition ensures that the input will have zero degree of membership in all other clusters. Using the FCM definition, the degree of fulfillment μ as a function of y is shown in Figure 3 for a typical rule. We see that μ is 1 when y is at the cluster center associated with the rule and drops sharply to zero as y approaches a neighboring cluster center. The output of the 0th order fuzzy model when the FCM definition of μ is adopted is shown in Figure 4. It is evident that the modeling accuracy has improved significantly; in particular, the model output trajectory is now compelled to pass through the cluster centers.

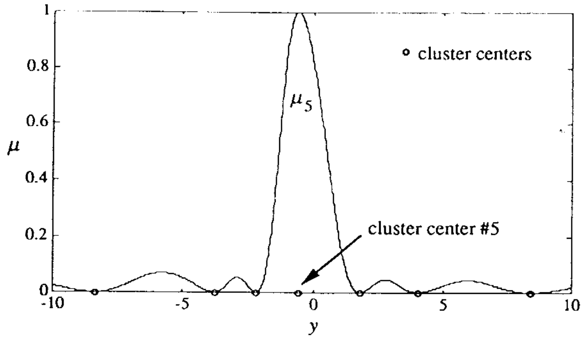

Figure 3. Degree of fillment of a typical rule as a function of input y, based on the fuzzy C-means definition of μ.

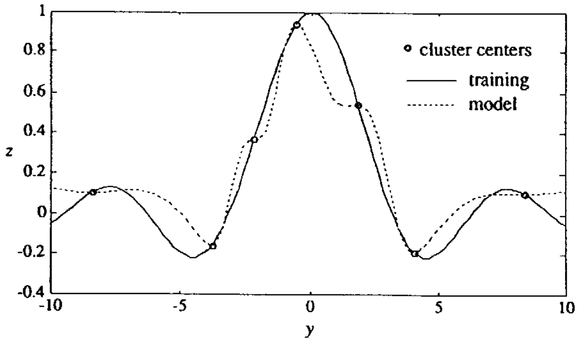

Figure 4. Comparison of training data with unoptimized 0th order model output, for the case where inference computation is based on the fuzzy C-means definition of μ.

Although using the FCM definition of μ can improve the accuracy of unoptimized 0th order models, we note that the μ function and the resultant model output trajectory have highly nonlinear "kinks" compared to that obtained with the exponential definition of μ. These kinks tend to limit the ultimately attainable accuracy when optimization is applied to the model.

We now apply least-squares optimization to z_i^*. For illustrative purposes, we consider the two cases: (1) $z_i^* = h_i$, and (2) $z_i^* = G_i y + h_i$. In the first case, we retain the assumption of a 0th order model, but now optimize the constant value assigned to z_i^*. In the second case, we assume a 1st order model and optimize both G_i and h_i. Table I shows the root-mean-square (RMS) modeling error resulting from the different extents of optimization. Table I also shows the effects of using the exponential definition of μ versus the FCM definition.

Table I. Comparison of RMS Modeling Error for Different Extents of Optimization and for Different Definitions of μ

Optimization	RMS error using EXP μ	RMS error using FCM μ
z_i^* unoptimized	0.180	0.119
$z_i^* = h_i$	0.054	0.098
$z_i^* = G_i y + h_i$	0.010	0.015

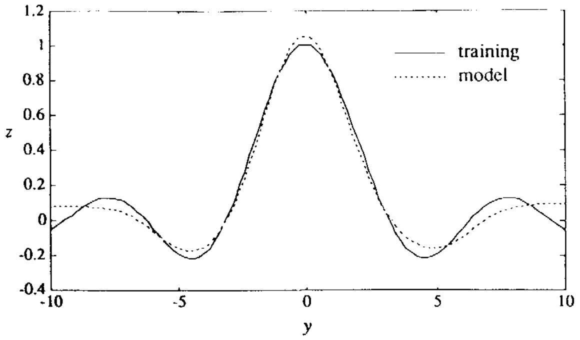

Figure 5. Comparison of training data with optimized 0th order model output.

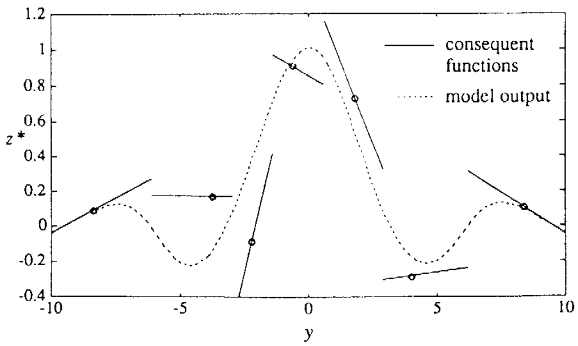

Figure 7. Consequent functions for the 1st order model.

Using the exponential definition of μ generally results in more accurate optimized models. In what follows, we will not draw any further comparisons between using the exponential definition versus using the FCM definition, but present only the results obtained from using the exponential definition. The output of the optimized 0th order model is shown in Figure 5 and the output of the optimized 1st order model is shown in Figure 6. The consequent function $z^* = G_i y + h_i$ of each rule is shown in Figure 7.

Although our method can be used to approximate a function from uniformly distributed data points as in the above example, the method is best suited for identifying models from repetitious experimental data, where there are repetitive behavior patterns that create distinct clusters in the data space. In the next example, we will examine a chaotic time series that does create such a data set.

Chaotic Time Series Prediction

We now consider a benchmark problem in model identification—that of predicting the time series generated by the chaotic Mackey–Glass differential delay equation (Mackey and Glass, 1977) defined by

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t)\,.$$

The task is to use past values of x up to the time t to predict the value of x at some time $t + \Delta t$ in the future. The standard input for this type of prediction is N points in the time series spaced S apart, i.e., the input vector is $y = \{x(t-(N-1)S), \ldots, x(t-2S), x(t-S), x(t)\}$. To allow comparison with the published results of other methods, we use $\tau = 17$, $N = 4$, $S = 6$, $\Delta T = 6$. Therefore, each data point in the training set consists of

$$x = \{x(t-18), x(t-12), x(t-6), x(t), x(t+6)\}$$

where the first 4 elements correspond to the input variables and the last element corresponds to the output variable. We will compare the performance of the model identification method with the Adaptive-Network-Based Fuzzy Inference System (ANFIS) proposed by Jang (1993) as well as other neural network-based and polynomial-fitting methods reported by Crowder (1990). The ANFIS algorithm also produces fuzzy models consisting of the Takagi–Sugeno type rules. After specifying the number of membership functions for each input variable, the ANFIS algorithm iteratively learns the parameters of the premise membership functions via backpropagation and optimizes the parameters of the consequent equations via linear least-squares estimation. ANFIS has the advantage of being significantly faster and more accurate than many pure neural network-based methods. Fast computation speed is attained by requiring much

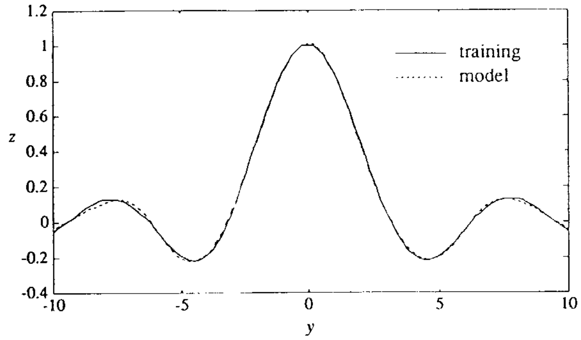

Figure 6. Comparison of training data with optimized 1st order model output.

less tunable parameters than traditional neural networks to achieve a highly nonlinear mapping, and is also attained by optimizing a large fraction of the parameters via linear least-squares estimation, thus further reducing the use of backpropagation. Because ANFIS typically has much less tunable parameters than a traditional neural network, it can avoid the pitfall of over-fitting the training data, thereby achieving excellent generalization ability. Comparison of ANFIS with our model identification method is particularly interesting because of the similarity in model structure. The performance of ANFIS provides a good indicator of the added benefit and computational burden that accompany nonlinear optimization of the Takagi–Sugeno type rules.

For the Mackey–Glass time series problem, we used the same data set as that used in (Jang 1993), which consisted of 1000 data points extracted from $t = 118$ to $t = 1117$. The first 500 data points were used for training the model, and the last 500 data points were used for checking the generalization ability of the model. As mentioned previously, the cluster radius r_a is an approximate specification of the desired resolution of the model, which can be adjusted based on the resultant complexity and generalization ability of the model. To illustrate this principle, we applied the cluster estimation method with different values of r_a ranging from 0.15 to 0.5 (i.e., the cluster radius was varied from 0.15 to 0.5 times the width of the data hypercube). This produced models of varying size, ranging from 69 rules to 9 rules. For each model, the consequent equations for a 1st order model were then optimized. Figure 8 shows the number of rules generated, as well as the modeling errors with respect to the training data and checking data, as functions of the cluster radius.

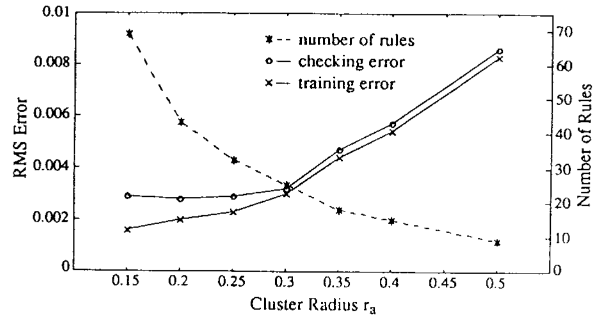

Figure 8. Model size and prediction error as functions of cluster radius.

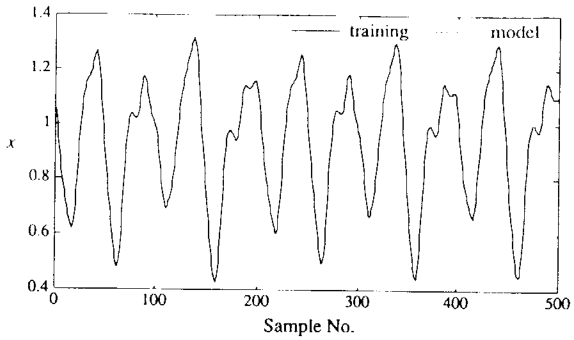

Figure 9. Comparison of training data with model output for Mackey–Glass time series.

We see that the error with respect to the training data and error with respect to the checking data begin to diverge when the cluster radius is less than 0.3, showing that the model is over-fitting the training data as the number of fitting parameters becomes too large. We use the results at $r_a = 0.3$ (a model with 25 rules) as the basis for comparison with other algorithms. Figures 9 and 10 show the model output evaluated with respect to training data and checking data, respectively. We see that the model output is indistinguishable from both training and checking data. The cluster centers and consequent equations for this model are listed in Appendix A.

The modeling error with respect to the checking data is listed in Table II along with the results from the other methods as reported in Jang (1993) and Crowder (1990). The error index shown in Table II is a non-dimensional error defined as the RMS error divided by the standard deviation of the actual time series (Jang, 1993; Crowder, 1990). Comparison between the various methods shows that the cluster estimation-based method can provide approximately the same degree of accuracy as the more complex methods. Only ANFIS produced a re-

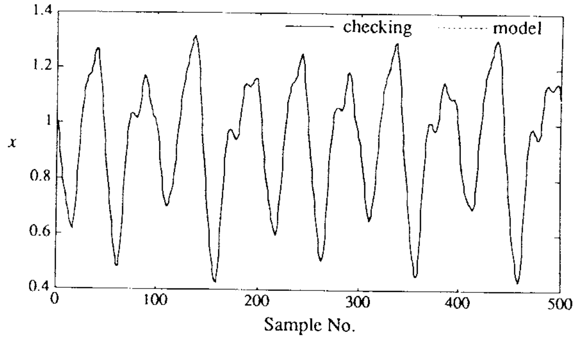

Figure 10. Comparison of checking data with model output for Mackey–Glass time series.

Table II. Comparison of Results from Different Methods of Model Identification*

Method	# Training Data	Error Index
Cluster Estimation-Based	500	0.014
ANFIS	500	0.007
Auto-Regressive Model	500	0.19
Cascaded-Correlation NN	500	0.06
Back-Prop NN	500	0.02
6th-order polynomial	500	0.04
Linear Predictive Method	2000	0.55

* Rows 2 and 3 are from Jang (1993); the last 4 rows are from Crowder (1990).

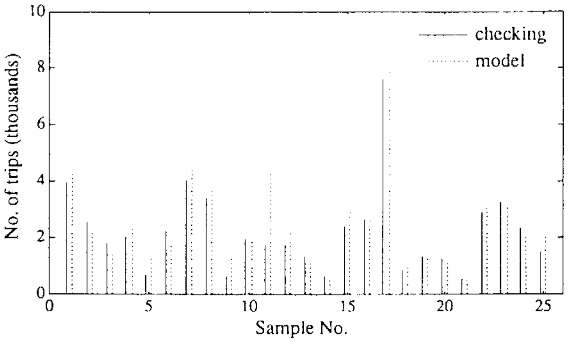

Figure 11. Comparison of checking data and model output for trip generation modeling.

sult that is more accurate than the cluster estimation-based method. The ANFIS model referenced here uses two membership functions for each input variable; for the 4-input Mackey–Glass problem, this leads to $2^4 = 16$ fuzzy partitions in the input space, and thus 16 rules. This ANFIS model has 24 parameters optimized via back-propagation (3 parameters for each premise membership function) and 80 parameters optimized via linear least-squares estimation (5 parameters for each consequent equation). The cluster estimation-based method does not involve any nonlinear optimization but the resultant model has 125 consequent parameters optimized via linear least-squares estimation. Identifying the Mackey–Glass model via the ANFIS algorithm (coded in C) required 1.5 hours on an Apollo 700 series workstation (Jang 1993), while the cluster estimation-based algorithm (also coded in C) was able to identify the model in 2 minutes on a Macintosh Centris 660AV (68040 processor running at 25 MHz).

Trip Generation Modeling

We have applied the model identification method to estimate the number of automobile trips generated from an area based on the demographics of the area. Five demographic factors were considered: population, number of dwelling units, vehicle ownership, median household income, and total employment. Hence, the model has 5 input variables and 1 output variable.

Demographic and trip data for 100 traffic analysis zones in New Castle County, Delaware, were used; this data was transcribed directly from (Kikuchi et al., 1994). Of the 100 data points, we randomly selected 75 as training data and 25 as checking data. Using $r_a = 0.5$, the model identification algorithm generated a 1st order model comprised of 3 rules. The computation time was 2 seconds on a Macintosh Centris 660AV. The cluster centers and consequent equations for this model are listed in Appendix B along with the data set. A comparison of the model output with that of the checking data is shown in Figure 11. The average modeling error with respect to the training data was 0.34 and that with respect to the checking data was 0.37, indicating the model generalizes well.

The fact that we can accurately cover a 6-dimensional data space with only 3 rules attests to the particular advantages of using the Takagi–Sugeno type rules.

Conclusion

We presented a cluster estimation method based on computing a measure of potential for each data point and iteratively reducing the potential of data points near new cluster centers. The computation grows only linearly with the dimension of the problem and as the square of the number of data points. This method can be used to estimate the number of clusters and their locations for initializing iterative optimization-based clustering algorithms such as fuzzy C-means, or it can be used as a stand-alone approximate clustering algorithm.

Combining the cluster estimation method with a linear least-squares estimation procedure provides a fast and robust algorithm for identifying fuzzy models from numerical data. Fast computation and robustness with respect to initial parameter values are achieved by avoiding any form of nonlinear optimization. Robustness with respect to noisy data is achieved by the data averaging that takes place in both the cluster estimation and least-squares estimation procedures. The cluster center selection criteria also avoid engendering a rule from a few erroneous outlying data points. Although there exist even simpler and faster model identification methods based on look-up tables (Wang and Mendel, 1992) and nearest neighbor clustering (Wang, 1993), they are sensitive to noisy data and prone to generating a rule from a single outlying data point. Compared with more complex model identification methods, our method can provide similar degree of accuracy and robustness with respect to noisy data while significantly reducing computational complexity.

The author thanks Jyh-Shing Jang at MathWorks Inc. for providing the data set for the Mackey–Glass benchmark problem.

Appendix A: Mackey–Glass Example

The input coordinates of the 25 cluster centers obtained in the Mackey–Glass example are

$$\begin{array}{llllll}
y_1^* = \{0.9479 & 1.0659 & 1.1352 & 1.1393\}; \\
y_2^* = \{1.0754 & 1.1344 & 1.1514 & 1.0481\}; \\
y_3^* = \{1.1302 & 1.1135 & 1.0133 & 0.7805\}; \\
y_4^* = \{0.6686 & 0.8264 & 1.0728 & 1.1912\}; \\
y_5^* = \{0.7423 & 0.6652 & 0.8227 & 1.0942\}; \\
y_6^* = \{1.1165 & 0.9671 & 0.7594 & 0.6748\}; \\
y_7^* = \{0.9206 & 0.7423 & 0.6652 & 0.8227\}; \\
y_8^* = \{0.4807 & 0.8575 & 1.0007 & 0.9736\}; \\
y_9^* = \{0.8636 & 0.9930 & 0.9652 & 1.1350\}; \\
y_{10}^* = \{1.2177 & 1.2001 & 0.8946 & 0.6462\}; \\
y_{11}^* = \{1.1578 & 1.2177 & 1.2001 & 0.8946\}; \\
y_{12}^* = \{0.5295 & 0.4807 & 0.8575 & 1.0007\}; \\
y_{13}^* = \{0.7295 & 0.4978 & 0.5256 & 0.9060\}; \\
y_{14}^* = \{0.7209 & 1.0240 & 1.1812 & 1.2551\}; \\
y_{15}^* = \{1.0503 & 0.7635 & 0.5383 & 0.5456\}; \\
y_{16}^* = \{1.0240 & 1.1812 & 1.2551 & 1.2063\}; \\
y_{17}^* = \{1.2885 & 1.0440 & 0.7295 & 0.4978\}; \\
y_{18}^* = \{0.8753 & 0.6038 & 0.4473 & 0.7307\}; \\
y_{19}^* = \{1.2064 & 1.2985 & 1.0902 & 0.7687\}; \\
y_{20}^* = \{1.2063 & 0.8753 & 0.6038 & 0.4473\}; \\
y_{21}^* = \{0.4542 & 0.6581 & 0.9713 & 0.9814\}; \\
y_{22}^* = \{1.0547 & 0.8067 & 0.7005 & 0.6865\}; \\
y_{23}^* = \{0.6581 & 0.9713 & 0.9814 & 1.0441\}; \\
y_{24}^* = \{0.8030 & 0.6702 & 0.6539 & 0.9690\}; \\
y_{25}^* = \{0.8792 & 1.1165 & 1.2329 & 1.3114\};
\end{array}$$

The corresponding output equations are

$$\begin{array}{lrrrl}
z_1^* = [-0.8697 & -0.8586 & 1.0985 & -0.0564]y + & 1.6473; \\
z_2^* = [-1.0967 & -0.9421 & 0.6509 & -0.6423]y + & 2.9419; \\
z_3^* = [-0.7946 & -1.0950 & 0.4198 & -0.1406]y + & 2.4973; \\
z_4^* = [\ 0.3083 & 0.4635 & -0.2907 & 0.4160]y + & 0.5212; \\
z_5^* = [\ 0.3139 & 0.3855 & -0.2474 & 1.0058]y + & -0.1857; \\
z_6^* = [-1.2992 & -1.1434 & 0.8059 & -0.1730]y + & 2.8149; \\
z_7^* = [-0.1268 & -0.1224 & 0.1161 & 0.3606]y + & 0.9253; \\
z_8^* = [-0.0703 & 0.2306 & 0.3994 & 0.3059]y + & 0.2751; \\
z_9^* = [\ 0.1789 & -0.8965 & 0.4342 & -0.7345]y + & 2.2661; \\
z_{10}^* = [-0.1002 & -0.7315 & 0.0730 & 0.5958]y + & 1.0337; \\
z_{11}^* = [-0.3041 & -0.5097 & -0.0372 & 0.3868]y + & 1.3105; \\
z_{12}^* = [\ 0.2663 & 0.4642 & -0.0061 & 0.6957]y + & -0.0812; \\
z_{13}^* = [\ 0.1212 & 0.7959 & -0.0774 & 0.6178]y + & -0.0061; \\
z_{14}^* = [\ 0.1507 & -0.8888 & 0.7914 & 0.5092]y + & 0.4449; \\
z_{15}^* = [-0.4307 & 0.2197 & -0.0546 & 0.4831]y + & 0.9631; \\
z_{16}^* = [\ 0.7240 & -0.8738 & 0.0097 & 0.9832]y + & -0.0493; \\
z_{17}^* = [\ 0.1229 & -1.4434 & 0.4796 & 0.6150]y + & 1.2162; \\
z_{18}^* = [\ 0.2107 & 0.7624 & -0.5270 & 0.8313]y + & -0.0301; \\
z_{19}^* = [\ 0.0484 & -0.7196 & 0.1410 & 0.7079]y + & 1.0026; \\
z_{20}^* = [-0.3792 & -0.8591 & 0.2753 & 0.4111]y + & 1.5885; \\
z_{21}^* = [\ 0.2770 & 0.4025 & 0.2096 & 0.5101]y + & -0.0530; \\
z_{22}^* = [-1.1732 & -0.3864 & 0.7371 & -0.2440]y + & 2.1758; \\
z_{23}^* = [-0.0953 & -0.1108 & 0.1202 & -0.1231]y + & 1.3095; \\
z_{24}^* = [\ 0.1712 & 0.6240 & -0.1140 & 0.7559]y + & -0.0701; \\
z_{25}^* = [\ 0.6685 & -1.9702 & 0.0110 & 1.3578]y + & 0.8858;
\end{array}$$

where y is a column vector of the input values: $[x(t-18), x(t-12), x(t-6), x(t)]$.

Appendix B: Trip Generation Modeling

The input coordinates of the 3 cluster centers obtained for the trip generation model are

$$\begin{array}{lrrrr}
y_1^* = \{1.5070 & 0.6570 & 0.7060 & 15.7350 & 0.6360\}; \\
y_2^* = \{3.1160 & 1.1930 & 1.4870 & 19.7330 & 0.6030\}; \\
y_3^* = \{0.0440 & 0.0240 & 0.0210 & 9.3400 & 0.8500\};
\end{array}$$

The corresponding output equations are

$$\begin{array}{lrrrr}
z_1^* = [\ 0.1337 & 0.1804 & -0.5726 & -0.0106 & 0.7147]y \\
\quad +\ 1.1966; \\
z_2^* = [-0.7630 & -0.1457 & 2.1074 & 0.0017 & 1.4631]y \\
\quad +\ 0.8378; \\
z_3^* = [-1.7168 & -1.9202 & 7.5741 & 0.0955 & 0.4707]y \\
\quad + -0.4319;
\end{array}$$

where y is a column vector of the input values: [population, number of dwelling units, vehicle ownership, median income, total employment]. The trip generation data are given in Table B-I. All numbers are expressed in units of a thousand.

Table B-I. Trip Generation Data

Population	Dwelling Units	Vehicle Ownership	Median Income	Total Employment	No. of trips
		Training Data			
0.038	0.032	0.019	11.429	16.439	8.460
0.148	0.062	0.051	9.375	3.176	2.250
0.244	0.139	0.068	6.673	0.193	0.317
0.222	0.204	0.064	7.738	3.450	1.907
0.023	0.013	0.039	10.125	7.064	4.746
0.411	0.192	0.093	7.824	0.179	0.375
0.132	0.040	0.017	5.378	0.627	0.540
0.044	0.024	0.021	9.340	0.850	0.768
0.066	0.011	0.004	11.705	0.859	0.605
0.537	0.250	0.203	17.406	2.568	2.298
1.465	0.471	0.436	17.340	1.341	1.708
1.357	0.433	0.303	8.260	0.411	0.843
4.071	1.348	1.339	11.590	1.486	2.801
2.277	0.789	0.569	18.451	0.602	1.239
3.837	1.355	1.347	14.125	1.077	2.756
1.590	0.762	0.747	11.703	0.357	1.304
3.116	1.193	1.487	19.733	0.603	2.385
1.368	0.517	0.708	25.681	1.265	1.883
0.937	0.448	0.639	21.691	0.585	1.242
3.419	2.333	1.766	13.403	0.870	2.946
1.877	0.763	0.917	18.750	1.565	2.183
1.872	0.831	1.065	27.661	0.667	1.868
3.107	1.369	1.561	16.843	1.146	2.772
1.535	0.605	0.784	16.544	0.399	1.313
2.326	0.931	0.612	11.454	0.573	1.472
3.340	0.925	0.775	11.560	0.942	1.852
2.223	0.850	0.970	16.400	2.524	3.177
3.274	1.065	1.003	13.193	0.750	2.037
1.490	0.592	0.686	13.796	1.797	2.037
3.447	1.366	1.396	16.461	0.705	2.349
3.035	1.176	1.494	18.963	0.447	2.169
0.003	0.004	0.002	6.036	1.687	1.272
1.600	0.506	0.260	5.369	1.385	1.467
0.398	0.151	0.132	8.159	0.625	0.648
2.396	0.901	0.542	8.603	1.270	1.838
2.698	1.268	0.541	6.663	0.758	1.548
0.414	0.144	0.080	13.119	0.245	0.387
0.037	0.015	0.005	9.659	1.277	0.966
2.760	1.333	1.200	13.498	1.064	2.996
4.476	1.656	2.719	26.266	0.817	3.631
1.507	0.657	0.706	15.735	0.636	1.446
1.179	0.434	0.744	32.328	1.353	2.490
1.727	0.690	1.232	30.573	1.397	3.253
3.118	1.175	1.995	30.063	0.568	3.400
3.103	1.147	2.198	28.859	2.115	5.584
1.057	0.535	0.639	16.101	0.371	1.521
1.250	0.490	0.731	21.040	0.449	1.169
0.759	0.368	0.440	13.719	2.780	3.043
1.219	0.331	0.630	30.581	0.087	0.929
3.918	1.414	2.196	21.240	0.389	3.059
2.413	1.004	1.407	20.289	0.675	2.309
2.487	0.783	1.514	24.101	1.399	3.194
1.089	0.430	0.734	18.971	0.247	1.114
0.293	0.117	0.188	27.220	0.108	0.518
2.101	0.836	1.318	18.686	0.322	2.105
1.028	0.453	0.657	15.944	1.812	4.431
3.562	1.360	1.844	20.053	2.000	5.049
4.247	1.651	2.526	20.093	0.265	3.138
2.674	1.022	1.649	26.034	0.109	2.015
1.381	0.515	0.961	31.250	3.363	2.302
1.131	0.494	0.731	18.873	1.286	1.714
2.210	0.805	1.460	25.000	0.477	2.383
0.450	0.188	0.212	13.251	0.418	0.951
3.206	1.075	1.567	17.099	0.710	2.373
3.071	1.035	1.562	20.454	0.599	2.670
2.358	0.732	1.235	24.159	0.592	1.749
2.507	0.857	1.495	21.843	0.073	1.723
6.577	2.122	3.710	26.345	0.536	4.902
3.047	1.175	2.014	25.889	0.684	3.151
3.551	1.674	2.131	19.401	1.584	4.698
0.229	0.098	0.146	30.428	1.927	5.127
2.731	0.811	1.545	34.528	2.473	6.575
3.510	1.098	2.273	41.679	0.235	2.864
3.390	1.073	2.329	48.440	0.316	3.074
2.246	0.809	1.477	31.303	0.111	1.803
		Checking Data			
4.107	1.389	2.623	24.234	0.789	3.930
2.057	0.715	1.359	36.643	0.487	2.530
1.641	0.739	1.149	23.285	0.473	1.809
0.035	0.015	0.012	12.624	3.239	2.017
0.803	0.290	0.523	28.835	0.131	0.674
3.457	1.237	1.604	17.665	0.269	2.229
4.940	1.760	3.161	22.458	0.647	4.058
3.605	1.098	2.288	37.019	0.611	3.390
0.118	0.048	0.013	15.753	0.752	0.652
2.336	0.790	1.461	22.426	0.385	1.944
3.492	1.515	2.296	21.174	0.983	1.746
1.047	0.238	0.419	17.173	1.264	1.737
1.239	0.504	0.792	22.173	0.222	1.322
0.110	0.082	0.028	7.786	0.689	0.641
2.264	0.866	1.661	35.994	0.431	2.384
2.637	0.897	1.680	27.938	0.357	2.660
4.783	1.737	2.784	22.288	3.490	7.605
1.093	0.323	0.695	30.884	0.019	0.860
1.859	0.765	0.733	14.185	0.396	1.346
2.043	0.542	0.292	6.386	0.777	1.251
0.384	0.128	0.083	9.401	0.423	0.529
2.042	0.759	1.471	25.619	1.259	2.911
3.219	1.337	1.782	15.466	0.795	3.221
2.280	0.681	1.391	27.095	0.521	2.327
0.942	0.296	0.372	19.811	1.137	1.465

References

Astrom K, Wittenmark B (1984): *Computer Controlled Systems: Theory and Design*, Englewood Cliffs, NJ: Prentice Hall.

Bezdek J (1974): "Cluster validity with fuzzy sets." *J. Cybernetics* **3**(3): 58–71.

Bezdek J, Hathaway R, Sabin M, Tucker W (1987): "Convergence theory for fuzzy c-means: Counterexamples and repairs." *The Analysis of Fuzzy Information*, Bezdek J (ed), CRC Press, Vol. 3, Chap. 8.

Crowder RS (1990): "Predicting the Mackey-Glass time series with cascade-correlation learning." In *Proc. 1990 Connectionist Models Summer School*, Carnegie Mellon University, pp. 117–123.

Dunn J (1974): "A fuzzy relative of the ISODATA process and its use in detecting compact, well separated cluster." *J. Cybernetics* **3**(3): 32–57.

Jang JSR (1993): "ANFIS: Adaptive-network-based fuzzy inference system." *IEEE Trans. on Systems, Man & Cybernetics* **23**(3): 665–685.

Kikuchi S, Nanda R, Perincherry V (1994): "Estimation of trip generation using the fuzzy regression method." 1994 Annual Meeting of Transportation Research Board, Washington, D.C.

Mackey M, Glass L (1977): "Oscillation and chaos in physiological control systems" *Science* **197**: 287–289.

Powell MJD (1987): "Radial basis functions for multivariable interpolation: A review." In *Algorithms for Approximation*, Mason J, Cox M (ed), Oxford: Clarendon Press, pp. 143–167.

Strobach P (1990): *Linear Prediction Theory: A Mathematical Basis for Adaptive Systems*, New York: Springer-Verlag.

Sugeno M, Tanaka K (1991): "Successive identification of a fuzzy model and its applications to prediction of a complex system." *Fuzzy Sets and Systems* **42**(3): 315–334.

Takagi T, Sugeno M (1985): "Fuzzy identification of systems and its application to modeling and control." *IEEE Trans. on Systems, Man & Cybernetics* **15**: 116–132.

Wang LX, Mendel JM (1992): "Generating fuzzy rules by learning from example." *IEEE Trans. on Systems, Man & Cybernetics* **22**(6).

Wang LX (1993): "Training of fuzzy logic systems using nearest neighborhood clustering." *Proc. 2nd IEEE Int'l Conf. on Fuzzy Systems (FUZZ-IEEE)*, San Francisco, CA, pp. 13–17.

Yager RR, Filev DP (1992): "Approximate clustering via the mountain method." Tech. Report #MII-1305, Machine Intelligence Institute, Iona College, New Rochelle, NY. Also to appear in *IEEE Trans. on Systems, Man & Cybernetics*.

Yager RR, Filev DP (1993): "Learning of fuzzy rules by mountain clustering." *Proc. SPIE Conf. on Applications of Fuzzy Logic Technology*, Boston, MA, pp. 246–254.

Received January 1994
Accepted June 1994