NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

Fuzzy Identification

Graduation Project
COM- 400

Student : Ahmad AL-zubi (990993)

Supervisor: Assoc. Prof. Dr. Rahib Abiyev

Nicosia - 2003

ACKNOWLEDGEMENTS

First of all I am happy to complete the task which I had given with blessing of God and also I am grateful to all the people in my life who have, supported me, advised me. Taught me and who have always encouraged me to follow my dreams and ambitions. My dearest parents, my brother and sisters, my friends and my tutors. They have taught me that no dream is unachievable. As in the words of Walt Disney "If you can dream it, you can do it."

I wish to thank my advisor, Assoc.Prof.Dr. Rahib Abiyev, for intellectual support, encouragement, and enthusiasm, which made this project possible, and his patience for correcting both my stylistic and scientific errors.

As I would like to thank the Near East University Stuff Professors, Assoc.Profs., Assist.Profs., Drs., Mrs., especially Prof.Dr. Fakhreddin Mamedov (Dean), Assoc.Prof.Dr. Adnan Khashman, Assoc.Prof.Dr. Dogan Ibrahim and Mr. Tayseer Alshanableh.

My sincerest thanks must go to my friends, Qais Albeni, Mohammad Ruzieh, Saleh Noures, Ghayth Al-najdawi, Mohammad Al-jabiri, Murad Al-zubi and Mohammad El-fawair who shared their suggestions and evaluations throughout the completion of my project. The comments from these friends enabled me to present this project successfully.

And above, I thank God for giving me stamina and courage to achieve my objectives.

AHMAD AL-ZUBI

ABSTRACT

In same technological processes are characterized by unpredictable and hand formulized factors, uncertainty and fuzziness of information. In this situation deterministic models is not enough adequately describe those processes and at the results control on their base begin difficult. In these conditions it is advisable to use fuzzy technology, which provide independency of the model to disturbance and adequacy of the model.

The aim of this project is identification of fuzzy model of technological process to solve this problem the state of application problem of fuzzy technology for identification of technological process is considered the methods which are use for identification problem are analyzed, the description of Least Squares Method and recursive least squares method, and to finding parameter of unknown system are described.

Also the description of Gradient Method its algorithm and application of TS system are considered parameters of updating formulas are presented.

Application of Clustering Method to identification problem is given.

Using above described algorithm, the identification of standard fuzzy system, TS fuzzy systems are given.

The application of fuzzy (LMS) method for modeling of fuzzy system is perform.

The implemental analysis and obtain result demonstrate the efficiency of application fuzzy technology for identification of system.

TABLE OF CONTENTS

INTRODUCTION

Fuzzy identification is an effective tool for the approximation of uncertain nonlinear systems on the basis of measured data. Data-driven identification techniques alone, however, sometimes yield unrealistic models in terms of steady-state characteristics, local linear behavior or physically impossible parameter values. This is typically due to insufficient information content of the identification data and due to over-parameterization of the models.

The Takagi-Sugeno (TS) fuzzy model is often used to represent nonlinear dynamic systems, by interpolating between local linear, time-invariant (LTI) ARX models. The TS fuzzy model is over-parameterized and when data-driven identification is used, the model can exhibit regimes, which are not found in the original system (Babuska, 1998). It is demonstrated in this paper that this problem can be remedied by incorporating prior knowledge into the identification method.

Recently, combinations of a priori knowledge with black-box modeling techniques have been gaining considerable interest. Two different approaches can be distinguished: gray-box modeling and semi-mechanistic modeling. In gray-box modeling, a priori information enters a black-box model, for instance, as constraints on the model parameters or variables, smoothness of the system behavior, or open-loop stability [18]. One can also start with deriving a model based on first principles and include black-box elements as parts of the white-box model. This approach is usually denoted as hybrid-modeling or semi-physical modeling [19].

In chapter [1] the state of application problems of fuzzy systems for identification of technological processes is considered.

In chapter [2] batch and recursive least squares methods for constructing a linear system to match some input-output data. Following this, we explain how these methods can be directly used for training fuzzy systems. We begin by discussing squares methods, as they are simple to understand and have clear connections to conventional estimation methods. We also present them first since they provide for the training of only certain parameters of a fuzzy system (e.g., the output membership function centers). Later, we will provide methods that can be used to tune al the fuzzy system's parameters.

And in chapter [3] show how gradient methods can be used to train a standard a Takagi-Sugeno fuzzy system. These methods are quite similar to the ones to train neural

network (e.g., the "back propagation techniques"). We provide examples for standard a Takagi-Sugeno fuzzy system. We highlight the fact that via either the recursive least squares method for fuzzy systems or the gradient method we can perform on-line parameter estimation.

And in chapter [4] two techniques for training fuzzy system based on clustering. The first uses "c-means clustering" and least squares to train the premises and consequents, respectively, of the Takagi-Sugeno fuzzy systems; while the second uses a nearest neighborhood technique to train standard fuzzy systems.

And in chapter [5] the analysis of complex systems, identification models. The TS type identification, fuzzy system modelings are described.

In conclusion the obtained important results and contribution is presented.

CHAPTER ONE

STATE OF APPLICATION PROBLEMS OF FUZZY TECHNOLOGY FOR IDENTIFICATION OF TECHOLOGICAL PROCESSES

'The closer one looks at a "real" world problem, the fuzzier becomes its solution. Stated informally, the essence of this principle is that, as the complexity of a system increases, our ability to make precise and yet significant statements about its behaviour diminishes until a threshold beyond which precision and significance (relevance) become almost mutually exclusive characteristics.' (L. Zadeh)

This throws light on the place of fuzziness in the models we wish to construct. The form of uncertainly handled by fuzzy models stems from the overall perception process of the system, and is caused by the complexity and level of knowledge of the system. In [16], the notation of the fuzzy system (model) is introduced, its motivation being as above. As argued in [13] and [14], fuzzy models are appropriate where goals and constraints, as well as the physical mechanisms present, are significant does not require from the model-builder (who may not possess the strict mathematical formulas of the process) a deep formal insight. But he usually has good intuition and a mature experience of the system.

Fuzzy set theory was found to be a very effective mathematical tool for dealing with the modeling and control aspects of complex industrial and not industrial processes as an alternative to other much more sophisticated mathematical models. Further, the latter circumstance led to the appearance at the beginning of the 1970's of fuzzy logic computer controllers which became a powerfully tool for coping with the complexity and uncertainty with which we are faced in many real-world problems of industrial process control. The first investigations in this field had to answer the question: Is it possible to realize a process controller which deals like a man with the involved linguistic information? The results of these inquires led to the design of the first fuzzy control systems which implemented in hardware and software a linguistic control algorithm. A control engineer on the base of the interviews then formulated such a control algorithm with human experts who currently work as process operators. The most simple fuzzy feedback control systems contain a fuzzy logic controller (FLC) in the form of a Table of linguistic rules, or fuzzy relation matrix and input-output interfaces.

Fuzzy logic has been successfully applied to many of industrial spheres, in robotics, in complex decision-making and diagnostic system, for data compression, in TV and others. Fuzzy sets can be used as a universal approximator, which is very important for modeling unknown objects.

There are number of examples about application of fuzzy logic to the modeling of different technological processes.

In [17] the development of a gray-box modeling approach for data-driven identification of dynamic Takagi–Sugeno (TS) fuzzy models is considered. The main idea is to constrain the candidate model parameters of the rules in the TS fuzzy model. Knowledge about the process such stability, minimal or maximal gains, or the settling time are translated into inequality constraints on the parameters. The fuzzy model then can be identified from input-output data by quadratic programming.

The proposed approach is applied to a laboratory liquid level process. A fuzzy model is first obtained from input-output measurements by using the proposed identification technique. The model is then used in model-based predictive control. Real-time control results show that when the gray-box identification algorithm is used, not only physically justified model is obtained, but also the performance of the model-based controller is improved with regard to the case where no prior knowledge is used.

In [20] a simplified incremental type of cause-effect models for additive MISO type dynamical processes is proposed and analyzed in this book. Dynamics of the process is expressed by three groups of parameters: gains, memory lengths and shapes of the specially introduced cause-effect relations membership functions. These functions represent in a fuzzy manner the degree of relationship between the past time changes of the respective input and the current change of the process output. The total model of the dynamical system in this book is identified from experimental data by different modifications of the Least Mean Squares algorithm for each group of parameters separately. The specially introduced indirect LMS algorithm is able to reduce significantly the size of the problem by identifying one-dimensional fuzzy model that represents indirectly the cause-effect relation for the dynamics. Several simulation examples are given as illustration and a brief analysis of the merits of the proposed algorithms for simulation and identification of real dynamical systems is made.

In [21] a study is described for several approaches to the identification of models for the temperature within the melter portion of a glass furnace. The focus is on developing models from the gas input to the throat (melter outlet) temperature.

Conventional linear techniques for system identification proved too useful as base-line comparisons for further studies involving nonlinear techniques from intelligent control for model building. Various combinations of input and output variables in a variety of model structure using fuzzy and neuro-fuzzy system modeling approaches are developed, and comparisons are drawn. Approaches reported on here investigate nonlinear Takagi-Sugeno (TS) fuzzy model formulations, where a linear-in-the-parameter identification problem is formulated for various combinations of measured variables and system delays. A fuzzy-neuro formulation is then discussed for parameter selection in the TS model structure while simultaneously optimizing the membership functions associated with the input of the TS fuzzy system. Simulation results for data collected from an operating glass furnace process are presented.

In [22] two different methods of fuzzy identification of a class of nonlinear systems are discussed in this book. This is applicable to systems with unknown and partially known mathematical models. The classes of systems considered are nonlinear in output but linear in input. In the first method, a gray box model is considered. The nominal values of parameters of the nonlinear system are assumed to be known. The unknown nonlinear function is identified off-line by choosing a suitable fuzzy relational model and the parameters of the nonlinear system are updated on-line using recursive least squares (RLS) algorithm. In the second method, a block box model is considered. The nonlinear plant is identified on-line by choosing a suitable linear model using RLS in stage-1 and the residual nonlinear part is identified in stage-2 using fuzzy identification. The control input is then calculated based on the identified nonlinear model using weighted one step a head control method.

In [23] the problem of identifying the parameters of the constituent local linear models of Takagi-Sugeno fuzzy models is considered. In order to address the tradeoff between global model accuracy and interpretability of the local models as linearizations of a nonlinear system, two multi-objective identification algorithms are studied. Particular attention is paid to the analysis of conflicts between objectives, and we show that such information can be easily computed from the solution of the multi-objective optimization. This information is useful to diagnose the model and tune the weighting/priorities of the multi-objective optimization. Moreover, the result of the conflict analysis can be used as a constructive tool to modify the fuzzy model structure (including membership functions) in order to meet the multiple objectives. The methods are illustrated on an experimental lungs respiration application.

In [24] approach for system identification among many others is the fuzzy identification approach. The advantage of this approach compared to the other analytical approaches is, that it is not necessary to make an assumption for the model to be used for the identification. In addition, the fuzzy approach can handle nonlinearities easier than analytical approaches. The Fuzzy-ROSA method is a method for databased generation of fuzzy rules. This is the first step of a two-step identification process. The second step is the optimization of the remaining free parameters, i.e., the composition of the rule base and the linguistic terms, to further improve the quality of the model and obtain small interpretable rule bases. In this book, a new evolutionary strategy for the optimization of the linguistic terms of the output variable is presented. The effectiveness of the two-step fuzzy identification is demonstrated on the benchmark problem 'Kin dataset' of the Delve dataset repository and the results are compared to analytical and neural network approaches.

CHAPTER TWO

LEAST SQUARES METHODS (LMS)

Many applications exist in the control and signal processing areas that may utilize nonlinear function approximation. One such application is system identification, which is the process of constructing a mathematical model of a dynamic system using experimental data from that system. Let g denote the physical system that we wish to identify. The training set G is defined by the experimental input-output data.

In linear system identification, a model is often used where

$$y(k) = \sum_{i=1}^{\bar{q}} \theta_{a_i} y(k-i) + \sum_{i=1}^{\bar{p}} \theta_{b_i} u(k-i)$$

and u(k) and y(k) are the system input and output at time $k \geq 0$. Notice that will need to specify appropriate initial conditions. In this case $f(x \mid \theta)$, which is not a fuzzy system, is defined by

$$f(x \mid \theta) = \theta^{\mathrm{T}} x$$

where

$$x(k) = [y(k-1), \cdots, y(k-\bar{q}), u(k), \cdots, u(k-\bar{p})]^{\mathrm{T}}$$

$$\theta = [\theta_{a_1}, \cdots, \theta_{a_{\bar{q}}}, \theta_{b_0}, \cdots, \theta_{b_{\bar{p}}}]^{\mathrm{T}}$$

Let $N = \bar{q} + \bar{p} + 1$ so that x(k) and θ are $N \times 1$ vectors. Linear system identification amounts to adjusting θ using information from G so that g(x) = $f(x \mid \theta)$ + e(x) where e(x) is small for all $x \in X$.

Similar to conventional linear system identification, for fuzzy identification we will utilize an appropriately defined "regression vector" x, and we will tune a fuzzy system $f(x \mid \theta)$ so that e(x) is small. Our hope is that since the fuzzy system $f(x \mid \theta)$ has more functional capabilities than the linear map, we will be able to achieve more accurate identification for nonlinear systems by appropriate adjustment of its parameters θ of the fuzzy system.

Next, consider how to view the construction of a parameter (or state) estimator as a function approximation problem. To do this, suppose for the sake of illustration that we seek to construct an estimator for a single parameter in a system g. Suppose further that we conduct a set of experiments with the system g in which we vary a parameter in the system say α. For instance, suppose we know that the parameter α lies in the range

$[\alpha_{\min}, \alpha_{\max}]$ but we do not know it lies and hence we would like to estimate it. Generate a data set G with data pairs $(x^i, \alpha^i) \in G$ where the α^i are a range of values over the interval $[\alpha_{\min}, \alpha_{\max}]$ and the x^i corresponding to each α^i is a set of input-output data from the system g that results from using α^i as the parameter value in g. Let $\hat{\alpha}$ denote the fuzzy system estimate of α. Now, if we construct a function $\hat{\alpha} = f(x \mid \theta)$ from the data in G, it will serves as an estimator for the parameter α. Each time a new x vectors is encountered, the estimator f will interpolate between the know associations $(x^i, \alpha^i) \in G$ to produce the estimate $\hat{\alpha}$. Clearly, if the data set G is "rich" enough, it will have enough (x^i, α^i) pairs so that when the estimator is presented with an $x \neq x^i$, it will have a good idea of what $\hat{\alpha}$ to specify because it will have many x^i that are close to x that it does know how to specify α for. We will study several applications of parameter estimation.

To apply function approximation to the problem pf how to construct a predictor for a parameter (or sate variable) in a system, we can proceed in a similar manner to how we did for the parameter estimation case above. The only significant difference lies in how to specify the data set G. In the case of prediction, suppose that we wish to estimate a parameter $\alpha\ (k+D)$, D times steps into the future. In this case we will need to have available training data pairs $(x^i, \alpha^i(k+D)) \in G$ that associate known future values of α with available data x^i. A fuzzy system constructed from such data will provide a predicated value $\hat{\alpha}\ (k+D) = f(x \mid \theta)$ for given values of x.

Overall, notice that in each case-identification, estimation, and prediction we rely on the existence of the data set G from which to construct the fuzzy system.

2.1 Batch Least Squares

We will introduce the batch least squares method to train fuzzy systems by first discussing the solution of the linear system identification problem. Let g denote the physical system that we wish to identify. The training set G is defined by the experimental input-output data that is generated from this system. In linear system identification, we can used model

$$y(k) = \sum_{i=1}^{\bar{q}} \theta_{a_i} y(k-i) + \sum_{i=0}^{\bar{p}} \theta_{b_i} u(k-i)$$

CHAPTER TWO

LEAST SQUARES METHODS (LMS)

2.1 Batch Least Squares

We will introduce the batch least squares method to train fuzzy systems by first discussing the solution of the linear system identification problem. Let g denote the physical system that we wish to identify. The training set G is defined by the experimental input-output data that is generated from this system. In linear system identification, we can used model

$$y(k) = \sum_{i=1}^{\bar{q}} \theta_{a_i} y(k-i) + \sum_{i=0}^{\bar{p}} \theta_{b_i} u(k-i)$$

where u(k) and y(k) are the system input and output at time k. in this case $f(x/\theta)$, which is not a fuzzy system, is defined by

$$f(x/\theta) = \theta^T x(k) \tag{2.1}$$

where we recall that

$$x(k) = [y(k-1),..., y(k-\bar{q}), u(k),..., u(k-\bar{p})]^T$$

and

$$\theta = [\theta_{a_1},...,\theta_{a_{\bar{q}}},\theta_{b_1},...,\theta_{b_{\bar{p}}}]^T$$

We have $N = \bar{q} + \bar{p} + 1$ so that x(k) and θ are $N \times 1$ vectors, and often x(k) is called the "regression vectors".

Recall that system identification amounts to adjusting θ using from G so that $f(x/\theta) \approx g(x)$ for all $x \in X$. Often, to form G linear system identification we choose $x^i = x(i), y^i = y(i)$, and let $G = \{(x^i, y^i) : i = 1,2,..., M\}$. To do this you will need appropriate initial conditions.

2.1.1 Batch Least Squares Derivation

In the batch least squares method we define

$$Y(M) = [y^1, y^2,..., y^M]^T$$

to be an $M \times 1$ vector of output data where the $y^i, i = 1,2,..., M$ come from $G(i.e., y^i$ such that $(x^i, y^i) \in G)$. We let

$$\Phi(M) = \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \\ (x^M)^T \end{bmatrix}$$

be an $M \times N$ matrix that consists of the x^i data vectors stacked into a matrix (*i.e.*, the x^i such that $(x^i, y^i) \in G$). Let

so that

$$E = Y - \Phi\theta$$

Choose

$$V(\theta) = \frac{1}{2} E^T E$$

to be a measure of how good the approximation is for all the data for a given θ. We want to pick θ to minimize $V(\theta)$. Notice that $V(\theta)$ is convex in θ so that a local minimum is a global minimum.

Now, using basic ideas from calculus, if we take the partial of V with respect to θ and set it equal to zero, we get an equation for $\hat{\theta}$, the best estimate (in the least squares sense) of the unknown θ. Another approach to deriving is to notice that

$$2V = E^T E = Y^T Y - Y^T \Phi\theta - \theta^T \Phi^T Y + \theta^T \Phi^T \Phi\theta$$

Then, we "complete the square" by assuming that $\Phi^T \Phi$ is invertible and letting

$$\begin{aligned} 2V = {} & Y^T Y - Y^T \Phi\theta - \theta^T \Phi^T Y + \theta^T \Phi^T \Phi\theta \\ & + Y^T \Phi(\Phi^T \Phi)^{-1} \Phi^T Y - Y^T \Phi(\Phi^T \Phi)^{-1} \Phi^T Y \end{aligned}$$

(where we are simply adding and subtracting the same terms at the end of the equation). Hence,

$$2V = Y^T (1 - \Phi(\Phi^T \Phi)^{-1} \Phi^T) Y + (\theta - (\Phi^T \Phi)^{-1} \Phi^T Y)^T \Phi^T \Phi(\theta - (\Phi^T \Phi)^{-1} \Phi^T Y)$$

The first term in this equation is independent of θ, so we cannot reduce V via this term, so it can be ignored. Hence, to get the smallest value of V, we choose θ so that the second term is zero. We will denote the value of θ that achieves the minimization of V by $\hat{\theta}$, and we notice that

$$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T Y \tag{2.2}$$

since the smallest we can make the last term in the above equation is zero. This is the equation for batch least squares that shows we can directly compute the least estimate $\hat{\theta}$

from the "batch" of data that is loaded into Φ and Y. If we pick the inputs to the system so that it is "sufficiently excited", then we will be guaranteed that $\Phi^T\Phi$ is invertible; if the data come from a linear plant with known $\bar{q}$ and $\bar{p}$, then for sufficiently large M we will achieve perfect estimation of the plant parameters.

In "weighted" batch least squares we use

$$V(\theta) = \frac{1}{2} E^T W E \tag{2.3}$$

where, for example, W is an $M \times M$ diagonal matrix with its diagonal elements $w_i > 0$ for $i = 1,2,...,M$ and its off-diagonal elements equal to zero. These w_i can be used to weight the importance of certain elements of G more than others. For example, we may choose to have it put less emphasis on older data by choosing $w_1 < w_2 < ... < w_M$ when x^2 is collected after x^1, x^3 is collected after x^2, and so on. The resulting parameter estimates can be shown to be given by

$$\hat{\theta}_{wbls} = (\Phi^T W \Phi)^{-1} \Phi^T W Y \tag{2.4}$$

To show this, simply use Equation (2.3) and proceed with the derivation in the same manner as above.

Example: Fitting a Line to Data

As an example of how batch least squares can be used, suppose that we would like to use this method to fit a line to a set of data. In this case our parameterized model is

$$y = x_1\theta_1 + x_2\theta_2 \tag{2.5}$$

Notice that if we choose $x_2 = 1$, y represents the equation for a line. Suppose that the data that we would like to fit the line to is given by

$$\left\{\left(\begin{bmatrix}1\\1\end{bmatrix},1\right),\left(\begin{bmatrix}2\\1\end{bmatrix},1\right),\left(\begin{bmatrix}3\\1\end{bmatrix},3\right)\right\}$$

Notice that to train the parameterized model in Equation (2.5) we have chosen $x_2^i = 1$ for $i = 1,2,3 = M$. We will use Equation (2.2) to compute the parameters for the line that best fits the data (in the sense that it will minimize the sum of the squared distances between the line and the data). To do this we let

$$\Phi = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$$

and

$$Y = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$$

Hence,

$$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T Y = \left(\begin{bmatrix} 14 & 6 \\ 6 & 3 \end{bmatrix} \right)^{-1} \begin{bmatrix} 12 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{1}{3} \end{bmatrix}$$

Hence, the line

$$y = x_1 - \frac{1}{3}$$

best the first data in the least squares sense. We leave it to the reader to plot the data points and this line on the same graph to see pictorially that it is indeed a good fit to the data.

The same general approach works for larger data sets. The reader may want to experiment with weighted batch least square to see how the weights w_i affect the way that the line will fit the data (making it more or less important that the data fit at certain points).

2.2 Recursive Least Squares

While the batch least squares approach has proven to be very successful for a variety of applications, it is by its very nature a "batch" approach (i.e., all the data are gathered, then processing is done). For small M we could clearly repeat the batch calculation for increasingly more data as they are gathered, but the computations became prohibitive due to the computation of the inverse of $\Phi^T \Phi$ and due to the fact that the dimensions of Φ and Y depend on M. Next, we derive a recursive version of the batch least squares method that will allow us to update our $\hat{\theta}$ estimate each time we get a new data pair, without using all the old data in the computation and without having to compute the inverse of $\Phi^T \Phi$.

Since we will be considering successively increasing the size of G, and we will assume that we increase the size by one each time step, we let a time k=M and i be such that $0 \le i \le k$. Let the $N \times M$ matrix

$$P(k) = (\Phi^T \Phi)^{-1} = \left(\sum_{i=1}^{k} x^i (x^i)^T \right)^{-1} \tag{2.6}$$

and let $\hat{\theta} = (k-1)$ denote the least squares estimate based on $k-1$ data pairs(P(k) is called the "covariance matrix"). Assume that $\Phi^T \Phi$ is nonsingular for all k. we have $P^{-1}(k) = \Phi^T \Phi = \sum_{i=1}^{k} x^i (x^i)^T$ so we can pull the last term from the summation to get

$$P^{-1}(k) = \sum_{i=1}^{k-1} x^i (x^i)^T + x^k (x^k)^T$$

and hence

$$P^{-1}(k) = P^{-1}(k-1) + x^k (x^k)^T \tag{2.7}$$

Now, using Equation (1.2) we have

$$\begin{aligned} \hat{\theta} &= (\Phi^T \Phi)^{-1} \Phi^T Y \\ &= \left(\sum_{i=1}^{k} x^i (x^i)^T \right)^{-1} \left(\sum_{i=1}^{k} x^i y^i \right) \\ &= P(k) \left(\sum_{i=1}^{k} {}^{i} y^i \right) \\ &= P(k) \left(\sum_{i=1}^{k-1} x^i y^i + x^k y^k \right) \end{aligned} \tag{2.8}$$

Hence,

$$\hat{\theta}(k-1) = P(k-1) \sum_{i=1}^{k-1} x^i y^i$$

and so

$$P^{-1}(k-1)\hat{\theta}(k-1) = \sum_{i=1}^{k-1} x^i y^i$$

Now, replacing $P^{-1}(k-1)$ in this equation with the result in Equation (2.7), we get

$$(P^{-1}(k) - x^k (x^k)^T)\hat{\theta}(k-1) = \sum_{i=1}^{k-1} x^i y^i$$

Using the result from Equation (2.8), this gives us

$$\begin{aligned} \hat{\theta}(k) &= P(k)(P^{-1}(k) - x^k (x^k)^T)\hat{\theta}(k-1) + P(k) x^k y^k \\ &= \hat{\theta}(k-1) - P(k) x^k (x^k)^T \hat{\theta}(k-1) + P(k) x^k y^k \end{aligned}$$

$$= \hat{\theta}(k-1+P(k)x^k(y^k-(x^k)^T\hat{\theta}(k-1)). \quad (2.9)$$

This provides a method to compute an estimate of the parameters $\hat{\theta}(k)$ at each time step k from the past estimate $\hat{\theta}(k-1)$ and the latest data pair that we received, (x^k, y^k). Notice that $(y^k-(x^k)^T\hat{\theta}(k-1))$ is the error in predicting y^k using $\hat{\theta}(k-1)$. To update $\hat{\theta}$ in Equation (1.9) we need $P(k)$, so we could use

$$P^{-1}(k) = P^{-1}(k-1) + x^k(x^k)^T \quad (2.10)$$

But then we will have to compute an inverse of a matrix at each step (i.e., each time we get another set of data). Clearly, this is not desirable for real-time implementation, so we would like to avoid this. To do so, recall that the "matrix inversion lemma" indicates that if A, C, and $(C^{-1}+DA^{-1}B)$ are nonsingular square matrices, then A+BCD is invertible A:

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1}$$

We will use this fact to remove the need to compute the inverse of $P^{-1}(k)$ that comes from Equation (1.10) so that can be used in Equation (1.9) to update $\hat{\theta}$. Notice that

$$\begin{aligned} P(k) &= (\Phi^T(k)\Phi(k))^{-1} \\ &= (\Phi^T(k-1)\Phi(k-1)+x^k(x^k)^T)^{-1} \\ &= (P^{-1}(k-1)+x^k(x^k)^T)^{-1} \end{aligned}$$

and that if we use the matrix inversion lemma with $A = P^{-1}(k-1), B = x^k, C = I$, and $D = (x^k)^T$, we get

$$P(k) = P(k-1) - P(k-1)x^k(I+(x^k)^T P(k-1)x^k)^{-1}(x^k)^T P(k-1) \quad (2.11)$$

which together with

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)x^k(y^k-(x^k)^T\hat{\theta}(k-1)) \quad (2.12)$$

(that was derived in Equation (1.9)) is called the "recursive least squares(RLS) algorithm." Basically, the matrix inversion lemma turns a matrix inversion into the inversion of a scalar (i.e., the term $(I+(x^k)^T P(k-1)x^k)^{-1}$ is a scalar).

We need initialize the RLS algorithm (i.e., choose $\hat{\theta}(0)$ and $P(0)$). One approach

to do this is to use $\hat{\theta}(0) = 0$ and $P(0) = P_0$ where $P_0 = \alpha$ I for some large $\alpha > 0$. This is the choice that is often used in practice. Other times, you may pick $P(0) = P_0$ but choose $\hat{\theta}(0)$ to be the best guess that you have at what the parameter values are.

There is a "weighted recursive least squares" (WRLS) algorithm also. Suppose that the parameters of the physical system θ vary slowly. In this case it may be advantageous to choose

$$V(\theta,k) = \frac{1}{2}\sum_{i=1}^{k} \lambda^{k-i}(y^i - (x^i)^T\theta)^2$$

where $0 < \lambda \leq 1$ is called a "forgetting factor" since it gives the more recent data higher weight in the organization (note that this performance index V could also be used to derive weighted batch least squares). Using a similar approach to the above, you can show that the equations for WRLS are given by

$$P(k) = \frac{1}{\lambda}(I - P(k-1)x^k(\lambda I + (x^k)^T P(k-1)x^k)^{-1}(x^k)^T)P(k-1) \qquad (2.13)$$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)x^k(y^k - (x^k)^T\hat{\theta}(k-1))$$

(where when $\lambda = 1$ we get standard RLS). This completes our description of the least squares methods. Next, we will discuss how they can be used to train fuzzy system.

2.3 Example: Batch least Squares Training of Fuzzy Systems

As an example of how to train fuzzy systems with batch least squares, we will consider how to tune the fuzzy system

$$f(x/\theta) = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}$$

(however, other forms may be used equally effectively). Here, b_i is the point in the output space at which the output membership function for the i^{th} rule achieves a maximum, c_j^i is the point in the j^{th} input universe of discourse where the membership function for the i^{th} rule achieves a maximum, and $\sigma_j^i > 0$ is the relative width of the membership function for the j^{th} input and the i^{th} rule. Clearly, we are using center-

average defuzzification and product for the premise and implication. Notice that the outermost input membership functions do not saturate, as is the usual case in control.

$$G = \left\{ \left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, 1 \right), \left(\begin{bmatrix} 2 \\ 4 \end{bmatrix}, 5 \right), \left(\begin{bmatrix} 3 \\ 6 \end{bmatrix}, 6 \right) \right\} \tag{2.14}$$

We will tune $f(x/\theta)$ to interpolate the data set G given in this Equation (2.14). Choosing R=2 and noting that n=2, we have $\theta = [b_1, b_2]^T$ and

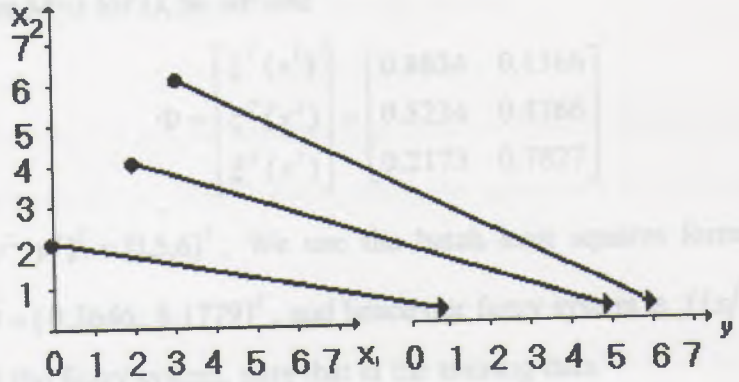


Figure 2.1 The training G generated from the function g.

$$\xi_i(x) = \frac{\prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i} \right)^2 \right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i} \right)^2 \right)} \tag{2.15}$$

Next, we must pick the input membership function parameters c_j^i, i=1,2, j=1,2. One way to choose the input membership function parameters is to use the x^i portions of the first R data pairs in G. In particular, we could make the premise of the rule i have unity certainty if $x^i, (x^i, y^i) \in G$, is input to the fuzzy system, $i = 1,2,...,R$, $R \le M$.

For instance, if $x^1 = [0,2]^T = [x_1^1, x_2^1]^T$ and $x^2 = [2,4]^T = [x_1^2, x_2^2]^T$, we would choose $c_1^1 = x_1^1 = 0, c_2^1 = 2$, $c_1^2 = x_1^2 = 2$, and $c_2^2 = x_2^2 = 4$.

Another approach to picking the c_j^i is simply to try to spread the membership functions somewhat evenly over the input portion of the training data space. For instance, consider the axes on the left of Figure 2.1 where the input portions of the training data are shown in G. From inspection, a reasonable choice for the input membership function centers could be $c_1^1 = 1.5, c_2^1 = 3, c_1^2 = 3$, and $c_2^2 = 5$ since this will

place the peaks of the primes membership functions in between the input portions of the training data pairs. In our examples, we will use this choice of the c_j^i.

Next, we need to pick the spreads σ_j^i. To do this we simply pick $\sigma_j^i = 2$ for $i = 1,2$, j=1,2 as a guess that we hope will provide reasonable overlap between the membership function. This completely specifies the $\xi_i(x)$ in Equation (2.15). Let $\xi(x) = [\xi_1(x), \xi_2(x)]^T$.

We have M=3 for G, so we find

$$\Phi = \begin{bmatrix} \xi^T(x^1) \\ \xi^T(x^2) \\ \xi^T(x^3) \end{bmatrix} = \begin{bmatrix} 0.8634 & 0.1366 \\ 0.5234 & 0.4766 \\ 0.2173 & 0.7827 \end{bmatrix}$$

and $Y = [y^1, y^2, y^3]^T = [1,5,6]^T$. We use the batch least squares formula in Equation (2.2) to find $\hat{\theta} = [0.3646,\ 8.1779]^T$, and hence our fuzzy system is $f(x/\hat{\theta})$.

To test the fuzzy system, note that at the training data

$$f(x^1/\hat{\theta}) = 1.4320$$
$$f(x^2/\hat{\theta}) = 4.0883$$
$$f(x^3/\hat{\theta}) = 6.4798$$

so that the trained fuzzy systems maps the training data reasonably ($x^3 = [3,6]^T$). Next, we test the fuzzy system at some points not in the training data set to see how it interpolates. In particular, we find

$$f([1,2]^T/\hat{\theta}) = 1.8267$$
$$f([2.5,5]^T/\hat{\theta}) = 5.3981$$
$$f([4,7]^T/\hat{\theta}) = 7.3673$$

These values seem like good interpolated values considering Figure (2.1), which illustrates the data set G for this example.

2.4 Examples: Recursive Least Squares Training of Fuzzy Systems

Here, we illustrate the use of the RLS algorithm in Equation (2.13) for training a fuzzy system to map the training data given in G in Equation (2.14). First, we replace x^k with $\xi(x^k)$ in Equation (2.13) to obtain

$$P(k) = \frac{1}{\lambda}(I - P(k-1)\xi(x^k)(\lambda \mathrm{I} + (\xi(x^k))^T P(k-1)\xi(x^k))^{-1}(\xi(x^k))^T P(k-1)$$
$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)\xi(x^k)(y^k - (\xi(x^k))^T \hat{\theta}(k-1)) \tag{2.15}$$

and we use this to compute the parameter vector of the fuzzy system. We will train the same fuzzy system that we considered in the batch least squares example of the previous section, and we pick the same c_j^i and σ_j^i, i = 1,2, j = 1,2 as we chose there so that we have the same $\xi(x) = [\xi_1, \xi_2]^T$.

For initialization of Equation (2.19), we choose

$$\hat{\theta}(0) = [2,5.5]^T$$

as a guess of where the output membership function centers should be. Another guess would be to choose $\hat{\theta}(0) = [0,0]^T$. Next, using the guidelines for RLS initialization, we choose

$$P(0) = \alpha \mathrm{I}$$

where $\alpha = 2000$. We choose $\lambda = 1$ since we do not want to discount old data, and hence we use the standard (nonweighted) RLS.

Before using Equation (1.19) to find an estimate of the output membership function centers, we need to decide in what order to have RLS process the training data pairs $(x^i, y^i) \in G$. For example, you could just take three steps with Equation (2.15), one for each training don't air. Another approach would be to use each $(x^i, y^i) \in G$ N_i times (in some order) in Equation (2.19) then stop the algorithm. Still another approach would be to cycle through all the data (i.e., (x^1, y^1) first, (x^2, y^2) second, up until (x^M, y^M) then go back to (x^1, y^1) and repeat), say, N_{RLS} times. It is this last approach that we will use and we will choose $N_{RLS} = 20$.

After using Equation (2.19) to cycle through the data N_{RLS} times, we get the last estimate

$$\hat{\theta}(N_{RLS}.M) = \begin{bmatrix} 0.3647 \\ 8.1778 \end{bmatrix} \tag{2.16}$$

and

$$P(N_{RLS}.M) = \begin{bmatrix} 0.0685 & -0.0429 \\ -0.0429 & 0.0851 \end{bmatrix}$$

Notice that the values produced for the estimates in Equation (2.16) are very close to the values we found with batch least squares-which we would expect since RLS is derived from batch least squares. We can test the resulting fuzzy system in the same way as we did for the one trained with batch least squares. Rather than showing the results, we simply note that since $\hat{\theta}(N_{RLS}.M)$ produced by RLS is very similar to the $\hat{\theta}$ produced by batch least squares, the resulting fuzzy system is quite similar, so we get very similar values for $f(x/\hat{\theta}(N_{RLS}.M))$ as we did for the batch least squares case.

CHAPTER THREE
GRADIENT METHODS

As in the previous sections, we seek to construct a fuzzy system $f(x \mid \theta)$ that can appropriately interpolate to approximate the function g that is inherently in the training data G. Here, however, we use a gradient optimization method to try to pick the parameters θ that perform the best approximation (i.e., make $f(x \mid \theta)$ as close to g(x) as possible). Unfortunately, while the gradient method tires to pick the best θ, just as for all the other method in this chapter, there are no guarantees that it will succeed in achieving the best approximation. As compared to the least squares methods, it does, however, provide a method to tune all the parameters of a fuzzy system. For instance, in addition to tuning the output membership function centers, using this method we can also tune the input membership function centers and spreads. Next, we derive the gradient training algorithms for both standard fuzzy systems and Takagi-Sugeno fuzzy systems that have only one output. In section 2.5 we extend this to the multi-input multi-output case.

3.1 Training Standard Fuzzy Systems

The fuzzy system used in this section utilizes singleton fuzzification, Gaussian input membership functions with centers c_j^i and spreads σ_j^i, output membership function centers b_i, product for the premise and implication, and center-average defuzzification, and we take on the form

$$f(x/\theta) = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)} \tag{3.1}$$

Note that we use Gaussian-shaped input membership functions for the entire input universe of discourse for all input and do not use ones that saturate at the outermost endpoints as we often do in control .The procedure developed below works in a similar fashion for other types of fuzzy system. Recall that c_j^i denotes the center for the i^{th} rule on the j^{th} universe of discourse, b_i denotes the center of the output

membership function for the i^{th} rule, and σ_j^i denotes the spread for the m^{th} training data pair $(x^m, y^m) \in G$. Let

$$e_m = \frac{1}{2}[f(x^m/\theta) - y^m]^2$$

In gradient methods, we seek to minimize e_m by choosing the parameters θ, which for our fuzzy system are b_i, c_j^i, and σ_j^i, $i = 1,2,...,R$, $j = 1,2,...,n$ (we will use $\theta(k)$ to denote these parameter's value at k). Another approach would be to minimize a sum of such error values for a subset of the data in G or all the data in G; however, with this approach computational requirements increase and algorithm performance may not.

3.1.1 Output Membership Function Centers Update Law

First, we consider how to adjust the b_i to minimize e_m. We use an "update law" (update formula).

$$b_i(k+1) = b_i(k) - \lambda_1 \frac{\partial e_m}{\partial b_i}\bigg|_k$$

where $i = 1,2,...,R$ and $k \geq 0$ is the index of the parameter update step. This is a "gradient descent" approach to choosing the b_i to minimize the quadratic function e_m that quantifies the error between the current data pair (x^m, y^m) and the fuzzy system. If e_m were quadratic in θ (which it is not), then this update method would move b_i along the negative gradient of the e_m error surface-that is, down the (we hope) bowl-shaped error surface (think of the path you take skiing down a valley-the gradient descent approach takes a route toward the bottom of the valley). The parameter $\lambda_1 > 0$ characterizes the "step size".

It indicates how big a step to take down the e_m error surface. If λ_1 is chosen small, then b_i is adjusted very slowly. If λ_1 is chosen too big, convergence may come faster but you risk it stepping over the minimum value of e_m (and possibly never converging to a minimum). Some work has been done on adaptively picking the step size. For example, if errors are decreasing rapidly, take big steps, but if errors are decreasing slowly, take small steps. This approach attempts to speed convergence yet avoid missing a minimum.

Now, to simplify the b_i update formula, notice that using the chain rule from calculus

$$\frac{\partial e_m}{\partial b_i} = (f(x^m/\theta) - y^m)\frac{\partial f(x^m/\theta)}{\partial b_i}$$

so

$$\frac{\partial e_m}{\partial b_i} = (f(x^m/\theta) - y^m)\frac{\prod_j^n \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^R \prod_{j=1}^n \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i}{\sigma_j^i}\right)^2\right)}$$

For notational convenience let

$$\mu_i(x^m, k) = \prod_{j=1}^n \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i(k)}{\sigma_j^i(k)}\right)^2\right) \tag{3.2}$$

and let

$$\in_m (k) = f(x^m/\theta(k)) - y^m$$

Then we get

$$b_i(k+1) = b_i(k) - \lambda_1 \in_m (k)\frac{\mu_i(x^m, k)}{\sum_{i=1}^R \mu_i(x^m, k)} \tag{3.3}$$

the update equation for the $b_i, i = 1,2,..., R, k \geq 0$.

The other parameters in $\theta, c_j^i(k)$ and $\sigma_j^i(k)$, will also be update with gradient algorithm to try to minimize e_m, as we explain next.

3.1.2 Input Membership Function Centers Update Law

To train the c_j^i, we use

$$c_j^i(k+1) = c_j^i(k) - \lambda_2 \left.\frac{\partial e_m}{\partial c_j^i}\right|_k$$

where $\lambda_2 > 0$ is the step size (see the comments above on how to choose this step size), $i = 1,2,..., R, j = 1,2,..., n$, and $k \geq 0$. At time k using the chain rule,

$$\frac{\partial e_m}{\partial c_j^i} = \in_m (k)\frac{\partial f(x^m/\theta(k))}{\partial \mu_i(x^m, k)}\frac{\partial \mu_i(x^m, k)}{\partial c_j^i}$$

for $i = 1,2,...,R$, $j = 1,2,...,n$, and $k \geq 0$. Now

$$\frac{\partial f(x^m/\theta(k))}{\partial \mu_i(x^m,k)} = \frac{\left(\sum_{i=1}^{R} \mu_i(x^m,k)\right) b_i(k) - \left(\sum_{i=1}^{R} b_i(k)\mu_i(x^m,k)\right)}{\left(\sum_{i=1}^{R} \mu_i(x^m,k)\right)^2} \quad (1)$$

so that

$$\frac{\partial f(x^m/\theta(k))}{\partial \mu_i(x^m,k)} = \frac{b_i(k) - \partial f(x^m/\theta(k))}{\sum_{i=1}^{R} \mu_i(x^m,k)}$$

Also,

$$\frac{\partial \mu_i(x^m,k)}{\partial c_j^i} = \mu_i(x^m,k)\left(\frac{x_j^m - c_j^i(k)}{\left(\sigma_j^i(k)\right)}\right)$$

so we have an update method for the $c_j^i(k)$ for all $i = 1,2,...,R$, $j = 1,2,...,n$, and $k \geq 0$.

In particular, we have

$$c_j^i(k+1) = c_j^i(k) - \lambda_2 \in_m (k)\left(\frac{b_i(k) - f(x^m/\theta(k))}{\sum_{i=1}^{R} \mu_i(x^m,k)}\right)\mu_i(x^m,k)\left(\frac{x_j^m - c_j^i(k)}{\left(\sigma_j^i(k)\right)^2}\right) \quad (3.4)$$

for $i = 1,2,...,R$, $j = 1,2,...,n$, and $k \geq 0$.

3.1.3 Input Membership Function Spreads Update Law

To update the $\sigma_j^i(k)$ (spreads of the membership functions), we follow the same procedure as above and use

$$\sigma_j^i(k+1) = \sigma_j^i(k) - \lambda_3 \frac{\partial e_m}{\partial \sigma_j^i}\bigg|_k$$

where $\lambda_3 > 0$ is the step size, $i = 1,2,...,R$, $j = 1,2,...,n$, and $k \geq 0$. Using the chain rule, we obtain

$$\frac{\partial e_m}{\partial \sigma_j^i} = \in_m (k) \frac{\partial f(x^m/\theta(k))}{\partial \mu_i(x^m,k)} \frac{\partial \mu_i(x^m,k)}{\partial \sigma_j^i}$$

We have

$$\frac{\partial \mu_i(x^m,k)}{\partial \sigma_j^i} = \mu_i(x^m,k) \frac{\left(x_j^m - c_j^i(k)\right)^2}{\left(\sigma_j^i(k)\right)^3}$$

so that

$$\sigma_j^i(k+1) = \sigma_j^i(k) - \lambda_3 \in_m (k) \frac{b_i(k) - f(x^m/\theta(k))}{\sum_{i=1}^{R} \mu_i(x^m,k)} \mu_i(x^m,k) \frac{\left(x_j^m - c_j^i(k)\right)^2}{\left(\sigma_j^i(k)\right)^3} \qquad (3.5)$$

for $i = 1,2,...,R, j = 1,2,...,n$, and $k \geq 0$. This complete the definition of the gradient training method for the standard fuzzy system, to summarize, the equation for updating the parameters θ of the fuzzy system are Equation (3.3), (3.4), and (3.5).

Next, note that the gradient training method described above is for the case where we have Gaussian-shaped input membership functions. The update formulas would, of course, change if you were to choose other membership functions. For instance, if you use triangular membership, the update formulas can be developed, but in this case you will have to pay special attention to how to define the derivative at the peak of the membership function.

Finally, we would like to note that the gradient method could be used in either an off-or on-line manner. In other words, it can be used off-line to train fuzzy system identification, or it can be used on-line to train a fuzzy system to perform real-time parameter estimation.

3.2 Implementation Issues and Example

In this section we discuss several issues that you will encounter if you implement a gradient approach to training fuzzy systems. Also, we provide an example of how to train a standard fuzzy system.

3.2.1 Algorithm Design

There are several issues to address in the design of the gradient algorithm for training a fuzzy system. As always, the choice of the training data G is critical. Issues in the choice of the training data, are relevant here. Next, note that you must pick the number of inputs n to the fuzzy system to be trained and the number of rules R; the method does not add rules, it just tunes existing ones.

The choice of the initial estimate $b_i(0), c_j^i(0)$, and $\sigma_j^i(k)$ can be important. Sometimes picking the close to where they should be can help convergence. Notice, that you should not pick $b_i = 0$ for all $i = 1,2,...,R$ or the algorithm for the b_i will stay at zero for all $k \geq 0$. Your computer probably will not allow you to pick $\sigma_j^i(0) = 0$ since you divide by this number in the algorithm. Also, you may need to make sure that in

that in the algorithm $\sigma_j^i(k) \geq \overline{\sigma} > 0$ for some fixed scalar $\overline{\sigma}$ so that the algorithm does not tune the parameters of the fuzzy system so that the computer has to divide by zero (to do this, just monitor the $\sigma_j^i(k)$, and if there exists some k' where $\sigma_j^i(k') < \overline{\sigma}$). Let $\sigma_j^i(k') = \overline{\sigma}$). Notice that for our choice of input membership functions

$$\sum_{i=1}^{R} \mu_i(x^m, k) \neq 0$$

so that we normally do not have to worry about dividing by it in the algorithm.

Note that the above gradient algorithm is for one only data pair. That is, we could run the gradient algorithm for along time (i.e., many values of k) for only one data pair to try to train the fuzzy system to match that data pair very well. Then we can go to the next data pair in G, begin with the final computed values of b_i, c_i, and σ_j^i from the last data pair we considered as the initial values for this data pair, and run the gradient algorithm for as many steps as we would like for that data pair– and so on. Alternatively, we could cycle through the training data many times, taking one step with the gradient algorithm for each data pair. It is difficult to know how many parameter update steps should be made for each data pair and how to cycle through the data. It is generally the case, however, that if you use some of the data much more frequently than other data in G, then the trained fuzzy system will tend to be more accurate for that data rather than the data that was not used as many times in training. Some like to cycle through the data so that each data pair is visited the same number of times and use small step sizes so that the updates will not be too large in any direction.

Clearly, you must be careful with the choices for the $\lambda_i, i = 1,2,3$ step sizes as values for these that are too big can result in an unstable algorithm (i.e., θ values can oscillate or became unbounded), while values for these that are too small can result in very slow convergence. The main problem, however, is that in the general case there are no guarantees that the gradient algorithm will converge at all! Moreover, it can take a signification amount of training data and long training times to achieve good result. Generally, you can conduct some tests to see how well the fuzzy system is constructed by comparing how it maps the data pairs to their actual values; however, even if this comparison appears to indicate that the fuzzy system is mapping the data properly, there are no guarantees that it will "generalize" (i.e., interpolate) for data not in the training data set that it was trained with.

To terminate the gradient algorithm, you could wait until all the parameters stop moving or change very little over a series of update steps. This would indicate that the parameters are not being updated so the gradients must be small so we must be at a minimum of the e_m surface. Alternatively, we could wait until the e_m or $\sum_{m=1}^{M} e_m$ does not change over affixed number of steps. This would indicate that even if the parameter values are changing, the values of e_m is not decreasing, so the algorithm has found a minimum and it can be terminated.

Example

As an example, consider the data set G in Equation (2.14): we will train the parameters of the fuzzy system with R = 2 and n = 2. Choose $\lambda_1 = \lambda_2 = \lambda_3 = 1$. Choose

$$\begin{bmatrix} c_1^1(0) \\ c_2^1(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} \sigma_1^1(0) \\ \sigma_2^1(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, b_1(0) = 1$$

and

$$\begin{bmatrix} c_1^2(0) \\ c_2^2(0) \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \begin{bmatrix} \sigma_1^2(0) \\ \sigma_2^2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_2(0) = 5$$

In this way the two rules will begin by perfectly mapping the first two data pairs in G. The gradient algorithm has to tune the fuzzy system so that it will provide an approximation to the third data pair in G, and in doing this it will tend to somewhat degrade how well it represented the first two data pairs.

To train the fuzzy system, we could repeatedly cycle through the data in G so that the fuzzy system learns how to map the third data pair but does not forget how to map the
first two. Here, for illustrative purpose, we will simply perform one iteration of the algorithm for the b_i parameters for the third data pair. That is, we use

$$x^m = x^3 = \begin{bmatrix} 3 \\ 6 \end{bmatrix}, y^m = y^3 = 6$$

In this case we have

$$\mu_1(x^3, 0) = 0.000003724$$

and

$$\mu_2(x^3, 0) = 0.08208$$

so that $f(x^3/\theta(0)) = 4.99977$ and $\in_m (0) = -1.000226$. With this Equation (3.4), we find that $b_1(1) = 1.000045379$ and $b_2(1) = 6.0022145$. The calculations for the $c_j^i(1)$ and $\sigma_j^i(1)$ parameters, i=1,2, j=1,2, are made in a similar way, but using Equation (3.4) an (3.5), respectively.

Even with only one computation step, we see that the output centers $c_j^i(1)$, i=1,2, are moving to perform an interpolation that is more appropriate for the third data point. To see the notice that $b_2(1) = 6.0022145$ where $b_2(0) = 5.0$ so that the output center moved much closer to $y^3 = 6$.

To further study how the gradient algorithm works, we recommend that you write a computer program to implement the update formulas for this example. You may need to tune the λ_i and approach to cycling through the data. Then, using an appropriate termination condition (see the discussion above), stop the algorithm and test the quality of the interpolation by placing inputs into the fuzzy system and seeing if the outputs are good interpolated values (e.g., compare them to Figure 2.1). In the next section we will provide a more detailed example, but for the training of Takagi-Sugeno fuzzy systems.

3.3 Training Takagi-Sugeno Fuzzy Systems

The Takagi-Sugeno fuzzy system that we train in this section takes on the form

$$f(x|\theta(k)) = \frac{\sum_{i=1}^{R} g_i(x,k)\mu_i(x,k)}{\sum_{i=1}^{R} \mu_i(x,k)}$$

where $\mu_i(x,k)$ is defined in Equation (2.21) (of course, other definition are possible), $x = [x_1, x_2, ..., x_n]^T$, and

$$g_i(x,k) = a_{i.0}(k) + a_{i.1}(k)x_1 + a_{i.2}(k)x_2 + ... + a_{i.n}(k)x_n$$

(note that we add the index k since we will update $a_{i.j}$ parameters).

3.3.1 Parameter Update Formulas

Following the same approach as in the previous section, we need to update the $a_{i.j}$ parameters of the $g_i(x,k)$ functions and c_j^i and σ_j^i. Notice, however, that most of

the work is done since if in Equations (3.3) and (3.4) we replace $b_i(k)$ with $g_i(x^m,k)$, we get the update formulas for the c_j^i and σ_j^i for the Takagi-Sugeno fuzzy system.

To update the $a_{i.j}$ we use

$$a_{i.j}(k+1) = a_{i.j}(k) - \lambda_4 \frac{\partial e_m}{\partial a_{i.j}}\bigg|_k \tag{3.6}$$

when $\lambda_4 > 0$ is the step size. Notice that

$$\frac{\partial e_m}{\partial a_{i.j}} = \epsilon_m(k) \frac{\partial f(x^m/\theta(k))}{\partial g_i(x^m,k)} \frac{\partial g_i(x^m,k)}{\partial a_{i.j}}$$

for all i = 1,2,…,R, j = 1,2,…,n (plus j = 0) and

$$\frac{\partial f(x^m/\theta(k))}{\partial g_i(x^m,k)} = \frac{\mu_i(x^m,k)}{\sum_{i=1}^R \mu_i(x^m,k)}$$

for all i = 1,2,…,R. Also,

$$\frac{\partial g_i(x^m,k)}{\partial a_{i.0}(k)} = 1$$

and

$$\frac{\partial g_i(x,k)}{\partial a_{i.j}(k)} = x_j$$

for all j = 1,2,…,n and i = 1,2,..,R.

This gives the update formulas for all the parameters of the Takagi-Sugeno fuzzy system. In the previous section we discussed issues in the choice of the step sizes and initial parameter values, how to cycle through the training data in G, and some convergence issues. All of this discussion is relevant to the training of Takagi-Sugeno models. The training of more general functional fuzzy systems where the g_i take on more general forms proceeds in a similar manner. In fact, it is easy to develop the update formulas for any functional fuzzy system such that

$$\frac{\partial g_i(x^m,k)}{\partial a_{i.j}(k)}$$

can be determined analytically. Finally, we would note that Takagi-Sugeno or general functional fuzzy systems could be trained either off- or on-line.

Example

As an example, consider once again the data set G in Equation (2.14). We will train the Takagi-Sugeno fuzzy system with rules (R = 2) and n = 2 considered in Equation (2.16). We will cycle through the data set G 40 times (similar to how we did in the RLS example) to get the error between the fuzzy system output and the portions of the training data to decrease to some small value.

We use Equations (3.5), (3.3), and (3.4) to update the $a_{i,j}(k), c_j^i(k)$, and $\sigma_j^i(k)$ values, respectively, for all i = 1,2,…,R, j = 1,2,…,n, and we choose $\bar{\sigma}$ from the previous section to be 0.01. For initialization we pick $\lambda_4 = 0.01, \lambda_2 = \lambda_3 = 1, a_{i,j}(0) = 1$, and $\sigma_j^i = 2$ for all i and j, and $c_1^1(0) = 1.5, c_2^1(0) = 3, c_1^2(0) = 3$, and $c_2^2(0) = 5$. The step sizes were tuned a bit to improve convergence, but could probably be further tuned to improve it more. The $a_{i,j}(0)$ values are simply somewhat arbitrary guesses. The $\sigma_{i,j}(0)$ values seem like reasonable spreads considering the training data. The $c_j^i(0)$ values are the same ones used in the least squares example and seem like reasonable guesses since they try to speared the premise membership function peaks somewhat uniformly over the input portions of the training data. It is possible that better initial guess for the $c_j^i(0)$ and $\sigma_{i,j}(0)$; in some ways this would make the guess for the $a_{i,j}(0)$ more consistent with the other initial parameters.

By the time the algorithm terminates, the error between the fuzzy system output and the output portions of the training data has reduced to less than 0.125 but is still showing a decreasing oscillatory behavior. At algorithm termination (k = 119), the consequent parameters are

$$a_{1,0}(119) = 0.8740, a_{1,1}(119) = 0.9998, a_{1,2}(119) = 0.7309$$

$$a_{2,0}(119) = 0.7642, a_{2,1}(119) = 0.3426, a_{2,2}(119) = 0.7642$$

the input membership function centers are

$$c_1^1(119) = 2.1982, c_1^2(119) = 2.6379$$

$$c_2^1(119) = 4.2833, c_2^2(119) = 4.7439$$

and their spreads are

$$\sigma_1^1(119) = 0.7654, \sigma_1^2(119) = 2.6423$$

$$\sigma_2^1(119) = 1.2713, \sigma_2^2(119) = 2.6636$$

These parameters, which collectively we call θ, specify that final Takagi-Sugeno fuzzy system.

To test the Takagi-Sugeno fuzzy system, we use the training data and some other cases. For the training data points we find

$$f(x^1|\theta) = 1.4573$$

$$f(x^2|\theta) = 4.8463$$

$$f(x^3|\theta) = 6.0306$$

so that the trained fuzzy system maps the training data accurately. Next, we test the fuzzy system at some points not in the training data set to see how it interpolates. In particular, we find

$$f([1,2]^T|\theta) = 2.4339$$

$$f([2,5.5]^T|\theta) = 5.7117$$

$$f([4,7]^T|\theta) = 6.6997$$

These values seem like good interpolated values considering Figure 2.1, which illustrates the data set G for this example.

3.4 Momentum Term and Step Size

There is some evidence that convergence properties of the gradient method can sometimes be improved via the addition of a "momentum term" to each of the update laws in Equations (3.3), (3.4), and (3.5). For instance, we could modify Equation (3.3) to

$$b_i(k+1) = b_i(k) - \lambda_1 \left.\frac{\partial e_m}{\partial b_i}\right|_k + \beta_i(b_i(k) - b_i(k-1))$$

i = 1,2,...,R where β_i is the gain on the momentum term. Similar changes can be made to Equations (3.4) and (3.5). Generally, the momentum term will help to keep the updates moving in the right direction. It is a method that has found wide use in the training of neural networks.

While for some applications a fixed step size λ_i can be sufficient, there has been some work done on adaptively picking the step size. For example, if errors are decreasing rapidly, take big update steps, but if errors are decreasing slowly take small

steps. Another option is to try adaptively picking the λ_i step sizes so that they best minimize the error

$$e_m = \frac{1}{2}\left[f(x^m/\theta(k)) - y^m\right]^2$$

For instance, for Equation (3.3) you could pick at time k the step size to be λ_1^* such that

$$\frac{1}{2}\left[f\left(x^m\left|\left\{\theta(k): b_i(k) - \lambda_1^* \frac{\partial e_m}{\partial b_i}\Big|_k\right\}\right.\right) - y^m\right]^2 =$$

$$\min_{\lambda_1 \in [0,\overline{\lambda_1}]} \frac{1}{2}\left[f\left(x^m\left|\left\{\theta(k): b_i(k) - \lambda_1 \frac{\partial e_m}{\partial b_i}\Big|_k\right\}\right.\right) - y^m\right]^2$$

(where $\overline{\lambda_1} > 0$ is some scalar that is fixed a priori) so that the step size will optimize the reduction of the error. Similar changes could be made to Equation (3.4) and (3.5). A vector version of the statement of how to pick the optimal step size is given by constraining all the components of $\theta(k)$, not just the output centers as we do above. The problem with this and batch is that it adds complexity to the update formulas since at each step an optimization problem must be solved to find the step size.

3.5 Newton and Gauss-Newton Methods

There are many gradient-type optimization techniques that can be used to pick θ to minimize e_m. For instance, you could use Newton, quasi-Newton, Gauss-Newton, or Levenberg-Marquardt methods. Each of these has certain advantages and disadvantages and many deserve consideration for a particular application.

In this section we will develop vector rather than scalar parameter update laws so we define $\theta(k) = [\theta_1(k), \theta_2(k), ..., \theta_p(k)]^T$ to be a $P \times 1$ vector. Also, we provide this development for n input, $\overline{N}$ output fuzzy system so that $f(x^m|\theta(k))$ and y^m are both $\overline{N} \times 1$ vectors.

The basic form of the update using a gradient method to minimize the function

$$e_m(k/\theta(k)) = \frac{1}{2}\left|f(x^m/\theta(k)) - y^m\right|^2$$

(notice that we explicitly add the dependence of $e_m(k)$ on $\theta(k)$ by using this notation) via the choice of $\theta(k)$ is

$$\theta(k+1) = \theta(k) + \lambda_k d(k) \tag{3.7}$$

where d(k) is the $P \times 1$ descent direction, and λ_k is a (scalar) positive step size that can depend on time k (not to be confused with the earlier notation for the step sizes). Here, $|x|^2 = x^T x$. For the descent function

$$\left(\frac{\partial e_m(k/\theta(k))}{\partial \theta(k)}\right)^2 d(k) < 0$$

and if

$$\frac{\partial e_m(k/\theta(k))}{\partial \theta(k)} = 0$$

where "0" is a $P \times 1$ vector of zeros, the method does not update $\theta(k)$. Our update formulas for the fuzzy system in Equations (3.3), (3.4), and (3.5) use

$$d(k) = -\frac{\partial e_m(k/\theta(k))}{\partial \theta(k)} = -\nabla e_m(k/\theta(k))$$

(which is the gradient of e_m with respect $\theta(k)$) so they actually provide for a "steepest descent" approach (of course, Equations (3.3), (3.4), and (3.5) are scalar update laws each with its own step size, while Equation (3.7) is a vector update law with a single step size). Unfortunately, this method can sometimes converge slowly, especially if it gets on along, low slope surface.

Next, let

$$\nabla^2 e_m(k/\theta(k)) = \left[\frac{\partial^2 e_m(k/\theta(k))}{\partial \theta_i(k)\theta_j(k)}\right]$$

be the $P \times P$ "Hessian matrix", the elements of which are the second partials of $e_m(k/\theta(k))$ at $\theta(k)$. In "Newton's method" we choose

$$d(k) = -(\nabla^2 e_m(k/\theta(k))^{-1} \nabla e_m(k/\theta(k)) \tag{3.8}$$

provided that $\nabla^2 e_m(k/\theta(k))$ is appositive definite so that it is invertible. For a function $e_m(k/\theta(k))$ that is quadratic in $\theta(k)$, Newton's method provides convergence in one step; for some other functions, it can converge very fast. The price you pay for this convergence speed is computation of Equation (3.8) and the need to verify the existence of the inverse in that equation.

In "quasi-Newton methods" you try to avoid problems with existence and computation of the inverse in Equation (3.8) by choosing

$$d(k) = -\Delta(k)\nabla e_m(k/\theta(k))$$

where $\Delta(k)$ is a positive definite $P \times P$ matrix for all $k \geq 0$ and is sometimes chose to approximate $\left(\nabla^2 e_m(k/\theta(k))\right)^{-1}$ (e.g., in some cases by using only the diagonal elements of $\left(\left(\nabla^2 e_m(k/\theta(k))\right)^{-1}\right)$. If $\Delta(k)$ is chosen properly, for some applications much of the convergence speed of Newton's method that is used can be achieved.

Next, consider the Gauss-Newton method that is used to solve a least squares problem such as finding $\theta(k)$ to minimize

$$e_m(k/\theta(k)) = \frac{1}{2}\left|f(x^m/\theta(k)) - y^m\right|^2 = \frac{1}{2}\left|\in_m (k/\theta(k))\right|^2$$

where

$$\in_m (k/\theta(k)) = f(x^m/\theta(k)) - y^m = [\in_{m_1}, \in_{m_2}, \ldots, \in_{m_{\bar{N}}}]^T$$

First, linearized $\in_m (k/\theta(k))$ around $\theta(k)$ (i.e., use a truncated Taylor series expansion) to get

$$\bar{\in}_m (\theta/\theta(k)) = \in_m (k/\theta(k)) + \nabla \in_m (k/\theta(k))^T (\theta - \theta(k))$$

Here,

$$\nabla \in_m (k/\theta(k)) = [\nabla \in_{m_1} (k/\theta(k)), \nabla \in_{m_2} (k/\theta(k)), \ldots, \nabla \in_{m_{\bar{N}}} (k/\theta(k))]$$

is a $P \times \bar{N}$ matrix whose columns are gradient vectors

$$\nabla \in_{m_i} (k/\theta(k)) = \frac{\partial \nabla \in_{m_i} (k/\theta(k))}{\partial \theta(k)}$$

$i = 1,2,\ldots,\bar{N}$. Notice that

$$\nabla \in_m (k/\theta(k))^T$$

is the "Jacobian". Also note that the notation $\bar{\in}_m (\theta/\theta(k))$ is used to emphasize the dependence on both $\theta(k)$ and θ.

Next, minimize the norm of the linearized function $\bar{\in}_m (\theta/\theta(k))$ by letting

$$\theta(k+1) = \arg\min_{\theta} \frac{1}{2}\left|\bar{\in}_m (\theta/\theta(k))\right|^2$$

Hence, in the Gauss-Newton approach we update $\theta(k)$ to a value that will best minimize a linear approximation to $\bar{\in}_m (\theta/\theta(k))$. Notice that

$$\theta(k+1) = \arg\min_{\theta} \frac{1}{2} [|\in_m (k/\theta(k))|^2 + 2(\theta - \theta(k))^T (\nabla \in_m (k/\theta(k)) \in_m (k/\theta(k))$$
$$+ (\theta - \theta(k))^T \nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T (\theta - \theta(k))]$$
$$= \arg\min_{\theta} \frac{1}{2} [|\in_m (k/\theta(k))|^2 + 2(\theta - \theta(k))^T (\nabla \in_m (k/\theta(k)) \in_m (k/\theta(k))$$
$$+ \theta^T \nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T \theta - 2\theta(k)^T \nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T \theta$$
$$+ \theta(k)^T \nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T \theta(k)] \tag{3.9}$$

perform this minimization, notice that we have a quadratic function so we find

$$\frac{\partial[\cdot]}{\partial\theta} = \nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k)) + \nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T \theta$$
$$- \nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T \theta(k) \tag{3.10}$$

where $[\cdot]$ denotes the expression in Equation (3.9) in brackets multiplied by one hale. Setting this equal to zero, we get the minimum achieved at θ^* where

$$\nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T (\theta^* - \theta(k)) = -\nabla \in_m (k/\theta(k))^T \nabla \in_m (k/\theta(k))$$

or, if $\nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T$ is a invertible,

$$\theta^* - \theta(k) = -\left(\nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T\right)^2 \nabla \in_m (k/\theta(k)) \in_m (k/\theta(k))$$

Hence, the Gauss-Newton update formula is

$$\theta(k+1) = \theta(k) - \left(\nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T\right)^2 \nabla \in_m (k/\theta(k)) \in_m (k/\theta(k))$$

To avoid problems with computing the inverse, the method is often implemented as

$$\theta(k+1) = \theta(k) - \lambda_k \left(\nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T + \Gamma(k)\right)^{-1} \nabla \in_m (k/\theta(k)) \in_m (k/\theta(k))$$

where λ_k is appositive step size that can change at each time k, and $\Gamma(k)$ is a $P \times P$ diagonal matrix such that

$$\nabla \in_m (k/\theta(k)) \nabla \in_m (k/\theta(k))^T + \Gamma(k)$$

is positive definite so that it is invertible. In the Levenberg-Marquardt method you choose $\Gamma(k) = \alpha I$ where $\alpha > 0$ and I is the $P \times P$ identity matrix. Essentially, Gauss-Newton iteration is an approximation to Newton iteration so it can provide for faster convergence than, for instance, steepest descent, but not as fast as a pure Newton method; however, computations are simplified. Note, however, that for each iteration of the Gauss-Newton method (as it is started above) we must find the inverse of a $P \times P$

matrix; there are, however, methods in the optimization literature for copying with this issues.

Using each of the above methods to train a fuzzy system is relatively straightforward.

For instance, notice that many of the appropriate partial derivatives have already been found when we developed the steepest descent approach to training.

CHAPTER FOUR

CLUSTERING METHODS

"Clustering" is the portioning of data into subset or groups based on similarities between the data. Hence, we will introduce two methods to perform fuzzy clustering where we seek to use fuzzy sets to define soft boundaries to separate data into groups. The methods here are related to conventional ones that have been developed in the field of pattern recognition. We begin with a fuzzy "c-means" technique coupled with least squares to train Takagi-Sugeno fuzzy systems, and then we briefly study a nearest neighborhood method for training standard fuzzy systems. In the c-means approach, we continue in the spirit of the previous method in that we use optimization to pick the clusters and, hence, the premise membership function parameters. The consequent parameters are chosen using the weighted least squares approach developed earlier. The nearest neighborhood approach also uses a type of optimization in the construction of cluster centers and, hence, the fuzzy system.

4.1 Clustering with Optimal Output Predefuzzification

In this section we will introduce the clustering with optimal output predefuzzification approach to train Takagi-Sugeno fuzzy system. We do this via the simple example we have used in previous sections.

4.1.1 Clustering for Specifying Rule Premises

Fuzzy clustering is the partitioning of a collection of data into fuzzy subsets or " clusters" based- on similarities between the data and can be implemented using an algorithm called fuzzy c-means. Fuzzy c-means is an iterative algorithm used to find grades of membership μ_{ij} (scalars) and cluster centers $\underline{v}^j$ (vectors of dimension $n \times 1$) to minimize the objective function

$$J = \sum_{i=1}^{M} \sum_{j=1}^{R} (\mu_{ij})^m \left| x^i - \underline{v}^j \right|^2 \tag{4.1}$$

where m > 1 is a design parameter. Hence, M is the number of input-output data pairs in the training data set G, R is the number of clusters (number of rules) we wish to calculate, x^i for i = 1,...,M is the input portion of the input-output training data pairs,

$\underline{v}^j = [v_1^j, v_2^j, ..., v_n^j]^T$ for j = 1,…,R are the clusters centers, and μ_{ij} for i = 1,…,M and j = 1,…,R is the grade of membership of x^i in the j^{th} cluster. Also, $|x| = \sqrt{x^T x}$ where x is a vector. Intuitively, minimization of J results in cluster centers being placed to represent groups (clusters) of data.

Fuzzy clustering will be used to form the premise portion of the If-Then rules in the fuzzy system we wish to construct. The process of "optimal output predefuzzification " (least squares training for consequent parameters) is used to form the consequent portion of the rules. We will combine fuzzy clustering and optimal output predefuzzification to construct multi-input single-output systems. Repeating the process for each of the outputs can do extension of our discussion to multi-input single-output systems

In this section we utilize a Takagi-Sugeno fuzzy system in which the consequent portion of the rule-base is a function of the crisp inputs such that

$$\textbf{If } H^j \textbf{ Then } g_j(x) = a_{j,0} + a_{j,1}x_1 + ... + a_{j,n}x_n \tag{4.2}$$

Where n is the number of inputs and H^j is an input fuzzy set given by

$$H^j = \left\{(x, \mu_{H^j}(x)) : x \in \chi_1 \times ... \times \chi_n\right\} \tag{4.3}$$

where χ_i is the i^{th} universe of discourse, and $\mu_{H^j}(x)$ is the membership function associated with H^j that represents the premise certainty for rule j; and $g_j(x) = \underline{a}_j^T \hat{x}$ where $\underline{a}_j = [a_{j,0}, a_{j,1}, ..., a_{j,n}]^T$ and $\hat{x} = [1, x^T]^T$ where j = 1,…,R. The resulting fuzzy system is a weighted average of the output $g_j(x)$ for j = 1,…,R and is given by

$$f(x/\theta) = \frac{\sum_{j=1}^{R} g_j(x)\mu_{H^j}(x)}{\sum_{j=1}^{R} \mu_{H^j}(x)} \tag{4.4}$$

where R is the number of the rules-base. Next, we will use the Takagi-Sugeno fuzzy system model, fuzzy clustering, and optimal output defuzzification to determine the parameters $\underline{a}_j$ and $\mu_{H^j}(x)$, which define the fuzzy system. We will do this via a simple example.

Suppose we use the example data set in Equation (2.14) that has been used in the previous section. We first specify a "fuzziness factor" m > 1, which is a parameter that determines the amount of overlap of the clusters. If m > 1 is large, then points with less membership in the j^{th} cluster have less influence on the determination of the new

cluster centers. Next, we specify the number of clusters R we wish to calculate. The number of clusters R equals the number of rules in the rule-base and must be than or equal to the number of the data pairs in the training data set G (i.e., $R \leq M$). We also specify the error tolerance $\epsilon_c > 0$, which is the amount of error allowed in calculating the cluster centers. We initialize the cluster centers $\underline{v}_0^j$ via a random number generator so that each component of $\underline{v}_0^j$ is no larger (smaller) than the largest (smallest) corresponding component of the input portion of the training data. The selection of $\underline{v}_0^j$, although somewhat arbitrary, may affect the final solution.

For our simple example, we chosen m = 2 and R = 2, and let $\epsilon_c = 0.001$. Our initial cluster centers were randomly chosen to be

$$\underline{v}_0^1 = \begin{bmatrix} 1.89 \\ 3.76 \end{bmatrix}$$

and

$$\underline{v}_0^2 = \begin{bmatrix} 2.47 \\ 4.76 \end{bmatrix}$$

so that each component lies in between x_1^i and x_2^i for i = 1,2,3 (see the definition of G in Equation (2.14)).

Next, we compute the new cluster centers $\underline{v}^j$ based on the previous cluster centers so that the objective function in Equation (4.1) is minimized. The necessary conditions for minimizing J are given by

$$\underline{v}_{new}^j = \frac{\sum_{i=1}^{M} x^i (\mu_{ij}^{new})^m}{\sum_{i=1}^{M} (\mu_{ij}^{new})^m} \tag{4.5}$$

where

$$\mu_{ij}^{new} = \left[\sum_{k=1}^{R} \left(\frac{\left| x^i - \underline{v}_{old}^j \right|^2}{\left| x^i - \underline{v}_{old}^k \right|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{4.6}$$

for each i = 1,...,M and for each j = 1,2,...,R such that $\sum_{j=1}^{R} \mu_{ij}^{new} = 1$ (and $|x|^2 = x^T x$). In Equation (4.6) we see that it is possible that there exits an i = 1,...,M such that $\left| x^i - \underline{v}_{old}^j \right|^2 = 0$ for some j = 1,2,...,R. In this case the μ_{ij}^{new} is undefined. To fix this

problem, let μ_{ij} for all i be any nonnegative numbers such that $\sum_{j=1}^{R} \mu_{ij} = 1$ and $\mu_{ij} = 0$ if $\left|x^i - \underline{v}_{old}^j\right|^2 \neq 0$.

Using Equation (4.6) for our example with $\underline{v}_{old}^j = \underline{v}_0^j$, $j = 1,2$, we find that $\mu_{11}^{new} = 0.6729$, $\mu_{12}^{new} = 0.3271$, $\mu_{21}^{new} = 00.9197$, $\mu_{22}^{new} = 0.0803$, $\mu_{31}^{new} = 0.2254$, and $\mu_{32}^{new} = 0.7746$. We use these μ_{ij}^{new} from Equation (4.6) to calculate the new cluster centers

$$\underline{v}_{new}^1 = \begin{bmatrix} 1.366 \\ 3.4043 \end{bmatrix}$$

and

$$\underline{v}_{new}^2 = \begin{bmatrix} 2.5410 \\ 5.3820 \end{bmatrix}$$

using Equation (1.34).

Next, we compare the distance between the current cluster centers $\underline{v}_{new}^j$ and previous cluster centers $\underline{v}_{old}^j$ (which for the first step is $\underline{v}_0^j$). If $\left|\underline{v}_{new}^j - \underline{v}_{old}^j\right| < \epsilon_c$ for all j = 1,2,...,R then the cluster centers $\underline{v}_{new}^j$ accurately represent the input data, the fuzzy clustering algorithm is terminated, and we proceed on to optimal output defuzzification algorithm (see below). Otherwise, we continue to iteratively use Equation (4.5) and (4.6) until we find cluster centers $\underline{v}_{new}^j$ that satisfy $\left|\underline{v}_{new}^j - \underline{v}_{old}^j\right| < \epsilon_c$ for j = 1,2,...,R. For our example, $\underline{v}_{old}^j = \underline{v}_0^j$, and we see that $\left|\underline{v}_{new}^j - \underline{v}_{old}^j\right| = 0.6328$ for j = 1 and 0.6260 for j = 2. Both of these values are greater than ϵ_c, so we continue to update the cluster centers.

Proceeding to the next iteration, let $\underline{v}_{old}^j = \underline{v}_{new}^j$, j = 1,2,...,R from the last iteration, and apply Equation (4.5) and (4.6) to find $\mu_{11}^{new} = 0.8233$, $\mu_{12}^{new} = 0.1767$, $\mu_{21}^{new} = 0.7445$, $\mu_{22}^{new} = 0.2555$, $\mu_{31}^{new} = 0.0593$, and $\mu_{32}^{new} = 0.9407$ using the cluster centers calculated above, yielding the new cluster centers

$$\underline{v}_{new}^1 = \begin{bmatrix} 0.9056 \\ 2.9084 \end{bmatrix}$$

and

$$\underline{v}_{new}^{2} = \begin{bmatrix} 2.8381 \\ 5.7397 \end{bmatrix}$$

computing the distances between these cluster centers and the previous ones, we find that $\left|\underline{v}_{new}^{j} - \underline{v}_{old}^{j}\right| > \epsilon_c$, so the algorithm continues. It 14 iterations before the algorithm terminates. (i.e., before we have $\left|\underline{v}_{new}^{j} - \underline{v}_{old}^{j}\right| \leq \epsilon_c = 0.001$ for all j = 1,2,...,R). When it does terminate, name the final membership grade value μ_{ij} and cluster centers $\underline{v}^{j}$, i = 1,...,M, j = 1,...,R.

For our example, after 14 iterations the algorithm finds $\mu_{11} = 0.9994$, $\mu_{12} = 0.0006$, $\mu_{21} = 0.1875$, $\mu_{22} = 0.8125$, $\mu_{31} = 0.0345$, $\mu_{32} = 0.9655$,

$$\underline{v}^{1} = \begin{bmatrix} 0.0714 \\ 2.0725 \end{bmatrix}$$

and

$$\underline{v}^{2} = \begin{bmatrix} 2.5854 \\ 5.1707 \end{bmatrix}$$

Notice that the clusters have converged so that $\underline{v}^{1}$ is near $x^1 = [0,2]^T$ and $\underline{v}^{2}$ lies in between $x^2 = [2.4]^T$ and $x^3 = [3,6]^T$.

The final values of $\underline{v}^{j}$, j = 1,2,...,R, are used to specify the premise membership functions for the i^{th} rule. In particular, we specify the premise membership functions as

$$\mu_{H^i}(x) = \left[\sum_{k=1}^{R} \left(\frac{\left| x^i - \underline{v}^{j} \right|^2}{\left| x^i - \underline{v}^{k} \right|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{4.7}$$

j = 1,2,...,R where $\underline{v}^{j}$, j = 1,2,...,R are the cluster centers from the last iteration that uses Equation (4.5) and (4.6). It is interesting to note that for large values of m we get smoother (less distinctive) membership functions. This is the primary guideline to use in selecting the value of m; however, often a good first choice is m = 2. Next, note that $\mu_{H^i}(x)$ is a premise membership function that is different from any that we have considered. It is used to ensure certain convergence properties of the iterative fuzzy c-means algorithm described above. With the premise of the rule defined, we next specify the consequent portion.

4.1.2 Least Squares for Specifying Rule Consequents

We apply "optimal output predefuzzification" to the training data to calculate the function $g_j(x) = \underline{a}_j^T \hat{x}$, j = 1,2,...,R for each rule (i.e., each cluster center), by determining the parameters $\underline{a}_j$. There are two methods you can use to find the $\underline{a}_j$.

Approach 1: For each cluster center $\underline{v}^j$, we wish to minimize the squared error between the function $g_j(x)$ and the output portion of the training data pairs. Let $\hat{x}^i = [1,(x^i)^T]^T$ where $(x^i, y^i) \in G$. We wish to minimize the cost function J_j given by

$$J_j = \sum_{i=1}^{M} (\mu_{ij})^2 \left(y^i - (\hat{x}^i)^T \underline{a}_j\right)^2 \tag{4.8}$$

for each j = 1,2,...,R where μ_{ij} is the grade of membership of the input portion of the i^{th} data pair for the j^{th} cluster that resulted from the clustering algorithm after it converged, y^i is the output portion of the i^{th} data pair $d^{(i)} = (x^i, y^i)$, and the multiplication of $(\hat{x}^i)^T$ and $\underline{a}_i$ defines the output associated with the j^{th} rule for the i^{th} training data point.

Looking at Equation (4.8), we see that the minimization of J_j via the choice of the $\underline{a}_j$ is a weighted least squares problem. From Chapter 2 and Equation (2.2), the solution $\underline{a}_j$ for j = 1,2,...,R to the weighted least squares problem is given by

$$\underline{a}_j = (\hat{X}^T D_j^2 \hat{X})^{-1} \hat{X}^T D_j^2 Y \tag{4.9}$$

where

$$\hat{X} = \begin{bmatrix} 1 \cdots 1 \\ x^1 \cdots x^M \end{bmatrix}^T$$

$$Y = \left[y^1, \ldots, y^M\right]^T,$$

$$D_j^2 = \left(diag([\mu_{1j}, \ldots, \mu_{Mj}])\right)^2$$

For our example the parameters that satisfy the linear function $g_j(x) = \underline{a}_j^T \hat{x}^i$ for j = 1,2 such that J_j in Equation (4.8) is minimized were found to be $\underline{a}_1 = [3,2.999,-1]^T$ and $\underline{a}_2 = [3,3,-1]^T$, which are very close to each other.

Approach 2: As an alternative approach, rather than solving R least squares problems, one for each rule, we can use the least squares methods discussed in Section 1 to specify the consequent parameters of the Takagi-Sugeno fuzzy system. To do this, we simply parameterize the Takagi-Sugeno fuzzy system in Equation (4.4) in a form so that it is linear in the consequent parameters and of the form

$$f(x/\theta) = \theta^T \xi(x)$$

where θ holds all the $a_{i,j}$ parameters and ξ is specified in a similar manner to how we did in Section 2.3. Now, just as we did in Chapter 2.3, we can use batch or recursive least squares methods to find θ. Unless, we indicate otherwise, we will always use approach 1 in these chapters.

4.1.3 Testing the Approximator

Suppose that we use approach 1 to specify the rule consequents. To test how accurately the constructed fuzzy system represents the training data set G in Figure 2.1, suppose that we choose the test point x' such that $(x', y') \notin G$. Specifically, we choose

$$x' = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

We would expect from Figure 2.1 that the output of the fuzzy system would lie somewhere between 1 and 5. The output is 3.9999, so we see that the trained Takagi-Sugeno fuzzy system seems to interpolate adequately. Notice also that if we let $x = x^i$, i = 1,2,3 where $(x^i, y^i) \in G$, we get values very close to the y^i, i = 1,2,3, respectively. That is, for this example the fuzzy system nearly perfectly maps the training data pairs. We also note that if the input to the fuzzy system is $x = [2.5,5]^T$, the output is 5.5, so the fuzzy system seems to perform good interpolation near the training data points.

Finally, we note that the $\underline{a}_j$ will clearly not always be as close to each other as for this example. For instance, if we add the data pair $([4,5]^T, 5.5)$ to G (i.e., make M = 4), then the cluster centers converge after 13 iterations (using the same parameters m and $\in_c$ as we did earlier). Using approach 1 to find the consequent parameters, we get

$$\underline{a}_1 = [-1.458, 0.7307, 1.2307]^T$$

and

$$\underline{a}_2 = [2.999, 0.00004, 0.5]^T$$

for the resulting fuzzy system, if we let $x = [1,2]^T$ in Equation (4.4), we get an output value of 1.8378, so we see that it performs differently than the case for M = 3, but that it does provide a reasonable interpolated value.

4.2 Nearest Neighborhood Clustering

As with the other approaches, we want to construct a fuzzy estimation system that approximates the function g that is inherently represented in the training data set G. We use singleton fuzzification, Gaussian membership functions, product inference, and center-average defuzzification, and the fuzzy system that we train is given by

$$f(x \mid \theta) = \frac{\sum_{i=1}^{R} A_i \prod_{j=1}^{n} \exp\left[-\left(\frac{x_j - v_j^i}{2\sigma}\right)^2\right]}{\sum_{i=1}^{R} B_i \prod_{j=1}^{n} \exp\left[-\left(\frac{x_j - v_j^i}{2\sigma}\right)^2\right]} \tag{4.10}$$

where R is the number of clusters (rule), n is the number of inputs,

$$\underline{v}^j = [v_1^j, v_2^j, ..., v_n^j]^T$$

are the cluster centers, σ is a constant and is the width of the membership functions, and A_i and B_i are the parameters whose values will be specified below (to train a multi-output fuzzy system, simply apply the procedure to the fuzzy system that generates each output). From Equation (4.10), we see that the parameters vector θ is given by

$$\theta = [A_1, ..., A_R, B_1, ..., B_R, v_1^1, ..., v_n^1, ..., v_1^R, ..., v_n^R, \sigma]^T$$

and is characterized by the number of clusters (rules) R and the number of the inputs n. Next, we will explain, via a simple example, how to use the nearest neighborhood clustering technique to construct a fuzzy system by choosing the parameter vector θ. Suppose that n = 2, $X \subset \Re^2$, and $Y \subset \Re$, and that we use the training data set G in Equation (2.14). We first specify the parameter σ, which defines the width of the membership functions. A small σ provides narrow membership functions that may yield a less smooth fuzzy system mapping, which may cause fuzzy system mapping not to generalize well for the data points not in the training set. Increasing the parameter σ will result in a smoother fuzzy system mapping. Next, we specify the quantity $\in_f$, which characterizes the maximum distance allowed between each of the cluster centers. The smaller $\in_f$, the more accurate are the clusters that represent the function g. For our

example, we chose $\sigma = 0.3$ and $\in_f = 3.0$. Specifically, we set $A_1 = y^1$, $B_1 = 1$, and $v_j^1 = x_j^1$ for j = 1,2,…,n.

If we take our first data pair,

$$(x^1, y^1) = \left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}, 1 \right)$$

we get $A_1 = 1, B_1 = 1$, and

$$\underline{v}^1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

which forms our first cluster (rule) for $f(x \mid \theta)$. Next, we take the second data pair,

$$(x^2, y^2) = \left(\begin{bmatrix} 2 \\ 4 \end{bmatrix}, 5 \right)$$

and compute the distance between the input portion of the data pair and each of the R existing cluster centers. And let the smallest distance be $\mid x^i - \underline{v}^l \mid$ (i.e., the nearest cluster to x^i is $\underline{v}^l$) where $\mid x \mid = \sqrt{x^T x}$. If $\mid x^i - \underline{v}^l \mid < \in_f$, then we do not add any clusters (rules) to the existing system, but we update the existing parameters A_l and B_l for the nearest cluster $\underline{v}^l$ to account for the output portion y^i of the current input-output data pair (x^i, y^i) in the training data set G. Specifically, we let

$$A_l := A_l^{old} + y^i$$

and

$$B_l := B_l^{old} + 1$$

These values are incremented to represent adding the effects of another data pair to the existing cluster. For instance, A_l is incremented so that the sum in the numerator of Equation (4.10) is modified to include the effects of the additional data pair without adding another rule. The value of B_l is then incremented to represent that we have added the effects of another data pair (it normalizes the sum in Equation (4.10)). Note that we do not modify the cluster centers in this case, just the A_l and B_l values; hence we do not modify the premises (that are parameterized via the cluster centers and σ), just the consequents of the existing rule that the new data pair is closest to.

Suppose that $|x^i - \underline{v}^l| < \in_f$. Then we add an additional cluster (rule) to represent the (x^2, y^2) information about the function g by modifying the parameter vector θ and letting R = 2 (i.e., we increase the number of clusters (rules)), $v_j^R = x_j^2$ for j = 1,2,…,n, $A_R = y^2$, and $B_R = 1$. These assignments of variables represent the explicit addition of a rule to the fuzzy system. Hence, for our example

$$\underline{v}^2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, A_2 = 5, B_2 = 1$$

The nearest neighbor clustering technique is implemented by repeating the above algorithm until all of the M data pairs in G are used.

Consider the third data pair,

$$(x^3, y^3) = \left(\begin{bmatrix} 3 \\ 6 \end{bmatrix}, 6 \right)$$

We would compute the distance between the input portion of the current data pair x^3 and each of the R = 2 cluster centers and find the smallest distance $|x^3 - \underline{v}^l|$. For our example, what is the value of $|x^3 - \underline{v}^l|$ and which cluster center is closet? To test how accurately the fuzzy system f represents the training data set G, suppose that we choose a test point x' such that $(x', y') \notin G$. Specifically, we choose

$$x' = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

We would expect the output value of the fuzzy system for this input to lie somewhere between 1 and 5.

CHAPTER FIVE

FUZZY IDENTIFICATION MODELS

5.1 Standard Fuzzy Systems

First, we consider a fuzzy system

$$y = f(x/\theta) = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} \tag{5.1}$$

where $x = [x_1, x_2, ..., x_n]^T$ and $\mu_i(x)$ as the certainty of the premise of the i^{th} rule (it is specified via the membership functions on the input universe of discourse together with the choice of the method to use in the triangular norm for representing the conjunction in the premise). The $b_i, i = 1,2,...,R$, values are the centers of the output membership functions. Notice that

$$f(x/\theta) = \frac{b_1 \mu_1(x)}{\sum_{i=1}^{R} \mu_i(x)} + \frac{b_2 \mu_2(x)}{\sum_{i=1}^{R} \mu_i(x)} + ... + \frac{b_R \mu_R(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

and that if we define

$$\xi_i(x) = \frac{\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} \tag{5.2}$$

then

$$f(x/\theta) = b_1 \xi_1(x) + b_2 \xi_2(x) + ... + b_R \xi_R(x)$$

Hence, if we define

$$\xi(x) = [\xi_1, \xi_2, ..., \xi_R]^T$$

and

$$\theta = [b_1, b_2, ..., b_R]^T$$

then

$$y = f(x/\theta) = \theta^T \xi(x) \tag{5.3}$$

We see the form of the model to be tuned is in only a slightly different form from the standard least squares case in Equation (2.1). In fact, if the μ_i are given, then $\xi(x)$ is given so that it is in exactly the right form for use by the standard least squares methods since we can view $\xi(x)$ as a known regression vector. Basically, the training

data x^i are mapped into $\xi(x^i)$ and the least squares algorithms produce an estimate of the best centers for the output membership function centers b_i.

This means that either batch or recursive least squares can be used to train certain types of fuzzy systems (once that can be parameterized so that they are "linear in the parameters," as in Equation (5.3)). All you have to do is replace x^i with $\xi(x^i)$ in forming the Φ vector for batch least squares, and in Equation (2.13) for recursive least squares. Hence, we can achieve either on-or off-line training of certain fuzzy systems with least squares methods. If you have some heuristic ideas for the choice of the input membership functions and hence $\xi(x)$, then this method can, at times, be quite effective (of course any known function can be used to replace any of the ξ_i in the $\xi(x)$ vector). We have found that some of the standard choices for input membership functions (e.g., uniformly distributed ones) work very well for some applications.

5.2 Takagi-Sugeno Fuzzy Systems

It is interesting to note that Takagi-Sugeno Fuzzy Systems, can also be parameterized so that they are linear in parameter, so that they can also be trained with either batch or recursive least squares methods, in this case, if we can pick the membership functions appropriately (e.g., uniformly distributed ones), then we can achieve a nonlinear interpolation between the linear output functions that are constructed with least squares.

In particular, a Takagi-Sugeno Fuzzy System is given by

$$y = \frac{\sum_{i=1}^{R} g_i(x)\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

where

$$g_i(x) = a_{i,0} + a_{i,1}x_1 + \ldots + a_{i,n}x_n$$

Hence, using the same approach as for standard fuzzy systems, we note that

$$y = \frac{\sum_{i=1}^{R} a_{i,0}\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + \frac{\sum_{i=1}^{R} a_{i,1}x_1\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + \ldots + \frac{\sum_{i=1}^{R} a_{i,n}x_n\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

We see that the first term is the standard fuzzy system. Hence, use the $\xi_i(x)$ defined in Equation (1.15) and reduce $\xi(x)$ and θ to be

$$\xi(x) = [\xi_1(x), \xi_2(x), ..., \xi_R(x), x_1\xi_1(x), x_2\xi_2(x), ..., x_1\xi_R(x), ..., x_n\xi_1(x), x_n\xi_2(x), ..., x_n\xi_R(x)]^T$$

and

$$\theta = [a_{1,0}, a_{2,0}, .., a_{R,0}, a_{1,1}, a_{2,1}, ..., a_{R,1}, ..., a_{1,n}, a_{2,n}, ..., a_{R,n}]^T$$

so that

$$f(x/\theta) = \theta^T \xi(x)$$

represents the Takagi-Sugeno Fuzzy System, and we see that it too is linear in the parameters. Just as for a standard fuzzy system, we can batch or recursive least squares for training $f(x/\theta)$. To do this, simply pick (a prior) the $\mu_i(x)$ and hence the $\xi_i(x)$ vector, process the training data x^i where $(x^i, y^i) \in G$ through $\xi(x)$, and replace x^i with $\xi(x^i)$ in forming the Φ vector for batch least squares, or in Equation (2.13) for recursive least squares.

Finally, note that the above approach to training will work for any nonlinear that is linear in the parameters. For instance, if there are known nonlinearities in the system of the quadratic form, you can use the same basic approach as the one descried above to specify the parameters of consequent functions that are quadratic.

This paper addresses the identification of fuzzy models with the structure proposed by Takagi and Sugeno (1985). This fuzzy model consists of a set of rules of the following form:

$R_{i_1,...,i_n}$: **If** z_1 is A_{1,i_1} **and ... and** z_n is A_{n,i_n} **then** $y = f_{i_1,...,i_n}(z_1, \ldots, z_n)$,

Where π is the number of inputs $z = [z_1, ..., z_n]$ is a vector containing all the inputs of the fuzzy model and $A_{j,i_j}(z_j)$ is the i_j th antecedent fuzzy set for the j th input. The same symbol is used to denote a fuzzy set and its membership function. M_j. is the number of the fuzzy sets on the j th input domain.

$f_{i_1,...,i_n}(z)$ is a (crisp) consequent function. For a given input, z, the output of the fuzzy model, y, is inferred by computing the weighted average:

$$y = \frac{\sum_{i_1=1}^{M_1} \cdots \sum_{i_n=1}^{M_n} \beta_{i_1,...,i_n} f_{i_1,...,i_n}(z_1, \ldots, z_n)}{\sum_{i_1=1}^{M_1} \cdots \sum_{i_n=1}^{M_n} \beta_{i_1,...,i_n}}, \tag{5.4}$$

where the weight, $B_{i_1,...,i_n} > 0$, is the overall truth value of the $i_1,...,i_n$ th the rule calculated as:

$$\beta_{i_1,\ldots,i_n} = \prod_{j=1}^{n} A_{j,i_j}(z_j).$$

Triangular membership functions are used in this article to define the antecedent fuzzy sets as shown in Figure 5.1, where a_{j,i_j} denotes the cores of fuzzy sets defined by:

$$a_{j,i_j} = \text{core}(A_{j,i_j}(z_j)) = \{z_j | A_{j,i_j}(z_j) = 1\}.$$

The cores of the adjacent fuzzy sets determine the support of a set:

$$A_{j,i_j}(z_j) = \frac{z_j - a_{j,i_j-1}}{a_{j,i_j} - a_{j,i_j-1}}, \quad a_{j,i_j-1} \le z_j < a_{j,i_j}$$

$$A_{j,i_j}(z_j) = \frac{a_{j,i_j+1} - z_j}{a_{j,i_j+1} - a_{j,i_j}}, \quad a_{j,i_j} \le z_j < a_{j,i_j+1}$$

This ensures that the sum of the membership functions is equal to one. Constraints such as this one help to obtain an interpretable grid-type partitioning rule-bases. The consequent-estimation method presented in this paper is, however, independent of the membership functions used.

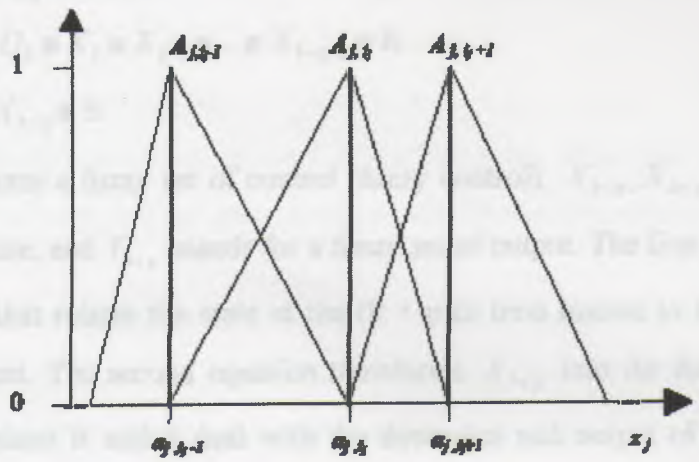


Fig.5.1 The membership functions used

As the product operator (3) is used for the 'and' connective, the overall truth values of the rules fulfill:

$$\sum_{i_1=1}^{M_1} \cdots \sum_{i_n=1}^{M_n} \beta_{i_1,\ldots,i_n} = 1.$$

Therefore, (5.4) can be simplified to:

$$y = \sum_{i_1=1}^{M_1} \cdots \sum_{i_n=1}^{M_n} \left[\left(\prod_{j=1}^{n} A_{j,i_j}(z_j) \right) f_{i_1,\ldots,i_n}(z_1,\ldots,z_n) \right] .$$

5.3 Identification of Fuzzy Models

5.3.1 Fuzzy Models

For fuzzy models, we use fuzzy-relational equations and a state-space methodology. Their generality is sufficient to allow us to analyze a variety system.

Let **X, U** and **Y** be the space of state, control and output, respectively. All of them are assumed to be finite, says $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$, $\mathbf{U} = \{u_1, u_2, \ldots, u_m\}$, $\mathbf{Y} = \{y_1, y_2, \ldots, y_v\}$. Obviously, the results obtained can be extended to models with spaces consisting of an infinite number of elements, but then we have to restrict ourselves to those t-norms for which infinite distributivity is preserved.

Those relationships govern the behavior of the system modeled using a set of fuzzy-relational equations:

$$X_{k+p} = U_k \blacksquare X_k \blacksquare X_{k+1} \blacksquare \ldots \blacksquare X_{k+p+1} \blacksquare \mathrm{R} \qquad (5.5)$$

$$Y_{k+p} = X_{k+p} \blacksquare \mathrm{S}$$

where U_k denotes a fuzzy set of control (fuzzy control), $X_{k+p}, X_{k+p-1}, \ldots, X_k$ refers to fuzzy sets of state, and Y_{k+p} stands for a fuzzy set of output. The first equation above is state equation that relates the state at the (k + p)th time instant to those occurring at previous moment. The second equation transforms X_{k+p} into the fuzzy set of output. The fuzzy relations R and S deal with the dynamics and output of the fuzzy model. They are defined in the Cartesian product of these spaces:

$$\mathrm{R}: U \times X \times \underbrace{X}_{-p-times} \times X - [0,1] \qquad , \qquad \mathrm{S}: X \times Y - [0,1]$$

If **X=Y** and S is an identity relation, S(x,y) = 1 if x = y and 0 otherwise, then the second equation of the model is irrelevant and only the state equation is of the interest. The model already is called a fuzzy model of the pth order and can be viewed as a generalized version of a difference equation, not necessarily a linear one, of the pth order. For p = 1, we arrive at a model of the first order:

$$X_{k+1} = U_k \blacksquare X_k \blacksquare \mathrm{R} \quad , \quad Y_{k+1} = X_{k+1} \blacksquare \mathrm{S} \tag{5.6}$$

With an extra assumption with respect to the output equation (S being the identity relation), this reduces to the single equation $X_{k+1} = U_k \blacksquare X_k \blacksquare \mathrm{R}$.

By propositions, we show how a model of the pth order can be transformed into a model of the first order. This enables us to concentrate on the last form of the model, i.e., that of the first order.

Proposition 5.1

The fuzzy model given by (5.5) can be reduced to a model of the first order:

$$\overline{X}_{k+1} = U_k \blacksquare X_k \blacksquare \overline{R} \quad , \quad \overline{Y}_{k+1} = \overline{X}_{k+1} \blacksquare \mathrm{S} \tag{5.7}$$

where R is defined in $U \times X \times X$, S in $X \times Y$, and X_{k+1} in X. To prove this, we define a fuzzy relation R resulting from a composition of the fuzzy sets of state and R:

$$\overline{R} = X_{k+1} \blacksquare X_{k+2} \blacksquare \ldots \blacksquare X_{k+p-1} \blacksquare \mathrm{R}$$

We write:

$$\overline{X}_{k+1} = X_{k+p} \quad , \quad \overline{Y}_{k+1} = Y_{k+p} \quad \text{and} \quad \overline{S} = S$$

Then, inserting all these into the original equations of the model, we obtain the reduced from of (5.7).

The use of this proposition is clear. In every case, we can omit reference to the method of reduction of fuzzy models of higher order. This also simplifies further consideration; without lack of generality, we can restrict ourselves to a model of the first order.

Now that we understand the structure of fuzzy models described by means of fuzzy-relational equation, it is worthwhile examining how the fuzziness processed in these models is handled. There are two sources of the fuzziness:

(i) First, in an intrinsic from of fuzziness, the ties between state and control variables are fuzzily known; i.e., one has only a fuzzy relation instead of a Boolean relation or function dependence. These models a part of the linguistic description of the system analyzed pertaining to a statement of the form:

'the state variables at consecutive discrete time intervals, say k and (k+1), are more or less equal if past control was quite high'

or

'at discrete time intervals, the state variable at time instant (k) is grater than in the past'

In both statements, there are elements of the linguistic description of the process. More precisely, there is a relation that is known only in an approximate way.

(ii) Secondly, an external form of fuzziness stems from the fact that we are dealing with linguistic values attached to the state and control variables, whereas the ties existing between them are known exactly. An example is the behavior of the pressure of an ideal gas in a container when its temperature and volume are specified only linguistically. Since we know the physical laws involved, the extension principle resolves the problem of the system description.

To cope with these two sources of fuzziness, we specify these formulas:

If only the intrinsic from of is present, U_k as well as X_k are degenerated fuzzy sets, with the membership functions defined as singletons, thus:

$$U_k(u) = \delta(u,u_k) \qquad , \qquad X_k = \delta(x,x_k)$$

Then X_{k+1} is equal to:

$$X_{k+1}(y) = \sup_{x,u}[U_k(u) t X_k(x) t R(u,x,y)]$$

$$= \sup_{x,u}[\delta_k(u,u_k) t \delta_k(x,x_k) t R(u,x,y)] = R(u_k,x_k,y)$$

Thus the fuzziness conveyed by the fuzzy relation of the model made the state model X_{k+1} fuzzy. Its membership function is equal to the membership function of the fuzzy relation, with their arguments $u = u_k$ and $x = x_k$ already determined by the non-fuzzy variables.

In the second case, where U_k and X_k are fuzzy sets and R is a function, the membership function is:

$$R(u,x,y) = \begin{cases} 1 & , \ if \quad y = f(u,x) \\ 0 & , \ otherwise \end{cases}$$

In other words, R(u,x,y) = δ(f(u,x),y). Performing formally the sup-t composition, we get:

$$X_{k+1}(y) = \sup[U_k(u) t X_k(x) t R(u,x,y)] = \sup_{u,x,y=f(u,x)} [U_k(u) t X_k(x)]$$

So we have the same result as we derived applying the extension principle to the fuzzy sets U_k and X_k transformed via the model equation y = f(u,x). If U_k and X_k are non-fuzzy, we have the well-known deterministic model of the system.

To allow fuzzy models formulated in the language of fuzzy-relational equations to be used effectively, we need to know how they are constructed and evaluated. To make fuzzy models applicable, we must solve the problem of system identification.

Nowadays, identification in deterministic (or statistical) models is a separate stream of engineering, with its own terminology and well-developed methodology. When studying fuzzy models, we are in a different position. Since they are quite novel concepts and too specific in comparison to the models studied (e.g., statistical models), we need to develop some new tools of analysis while keeping to the previous methodology. We develop this idea on system identification in the presence of fuzziness in the following sections.

5.3.2 System Identification in the Presence of Fuzziness

The methodology of system identification in this environment is not well established. Many fuzzy models are proposed, but they are given only ad-hoc consideration against an intuitive, vague background, without details of estimation procedures. What is worse, the parameters of the fuzzy models, which are treated as fuzzy sets (or fuzzy numbers), are accepted from model-builder without verification. It is assumed that the membership functions of the parameters in the model do not need to be estimated. This is unreal: intuition can help in creating the structure of a model, but one cannot expect to get the relevant membership functions of the model without performing any identification. As a loose analogy, consider regression analysis in constructing a model with one input and one output. Perhaps an assumption of liner dependence between the output and input variables may take sense in some cases where the model-builder has an intuitive view. Nevertheless, he cannot expect to get, even approximately, the parameters of the model. Despite a board variety of models, their identification procedures consist of these three clearly defined steps [1,2]:

1. determination of the structure of the model,
2. estimation of the parameters of the model from the data set provided for identification,
3. validation of the model by testing its consistency with the data set.

These steps are treated broadly: thus the choice of the data set, which is not a trivial task, should be part of the first step. We need to know not only the estimated parameters of the model but also they're precision, fundamental to the relevance of the

fuzzy model. For a deeper look at the identification problem, we examine the first step in more detail:

Determination of the structure of the fuzzy model

This step is crucial to further investigation. As the fuzzy models refer to a linguistic description for our process, we have conditional statements of this format: if, in the past k,k-1,..., $k-\ell$ time instants, the states of the system are given by fuzzy labels $X_k, X_{k+1}, \ldots, X_{k-\ell}$ and the control is equal to $U_{k-\ell}$, the state of the process is given by X_{k+1}.

The form of the fuzzy model directly implies this, with its order specified by the 'length' of these conditional statements. In fact, this length is not so large, since a conceptual model of the process realized by a human being consists of statements without a long condition part. Usually, a fuzzy model of low order is sufficient. In contrast, statistical models use sophisticated tools to estimate the order of the model equations and their type (i.e., linear or nonlinear). Working with fuzzy-relational equations, we are mostly concerned with the order of an equation rather than its type. The first notion is clear, while the second requires further optimization. The order of the model equation has a straightforward impact on the form of the data set collected for the estimation stage. Consider, for instance, a model governed by:

$$X_{k+2} = U_k \blacksquare X_k \blacksquare X_{k+1} \blacksquare \mathrm{R}$$

with U_k a fuzzy set of control, and X_k, X_{k+1}, X_{k+2} the fuzzy sets of state. The data set for determining the fuzzy relation R of the model consists of ordered triples. Starting with k = 1, we enumerate them thus:

$$U_1, X_1, X_2, X_3$$
$$\vdots$$
$$U_N, X_N, X_{N+1}, X_{N+2}$$

From Proposition 5.1, a model of higher order can be reduced to a model of the first order; therefore, we examine this equation:

$$X_{k+2} = U_k \blacksquare X_k \blacksquare \mathrm{R}$$

Now the relevant data set consists of triples:

$$U_1, X_1, X_2$$
$$\vdots$$
$$U_N, X_N, X_{N+1}$$

While considering estimation procedures, for notation simplicity we rewrite the model equation:

$$Y_k = X_k \blacksquare \mathrm{R} \tag{5.8}$$

which is equivalent to the previous one, since X_k now replaces the fuzzy relation $U_k \blacksquare X_k$:

$$(U_k \blacksquare X_k)(\mathrm{u,x}) = U_k(u) t X_k(x)$$

This simplifies the format of the data set, which now converts into pairs $(X_1,Y_1),(X_2,Y_2),\ldots,(X_N,Y_N)$. Thus the estimation problem is the task of solving a set of fuzzy relational equations $Y_k = X_k \blacksquare \mathrm{R}$ for R unknown.

Estimation of the fuzzy relation of the fuzzy model

In our first approach, we calculate the fuzzy relation of the model as an intersection of the partial results of the φ-compositions of X_k and Y_k:

$$\mathrm{R} = \bigcap_{k=1}^{N} (X_k \varphi Y_k) \tag{5.9}$$

This procedure consists of a sequence of steps:

1. put R = 1 (i.e., all the elements of R are set to 1.0),
2. put k = 1,
3. calculate $X_k \varphi Y_k$ and intersect R, $\mathrm{R} = \mathrm{R} \cap (X_k \varphi Y_k)$; increase k, k = k+1,
4. if $k \le N$, repeat (3); otherwise, stop.

Its output is the fuzzy relation of the fuzzy model. One remark should be made here. The fuzzy relation of the model computed using (5.8) assumes that the set of solutions of this family of equations is nonempty. The situation is made more difficult by the fact that this assumption is impossible to verify before the fuzzy relation of the model is calculated. This does not mean that this method of determining the relation should be discarded as theoretical but not practical. It can be used circumspectly, not overestimating the significance of its results and checking their validity carefully. Even if they are not satisfactory, they may give a good starting-point for applying more refined methods, to be clarified later. We call the method that utilizes the findings for solving fuzzy-relational equations, without checking whether a solution exists, a brute-force one.

The model may also be constructed using fuzzy-relational equations of different type, e.g. equations with inf-s composition, ad-joint equations or polynomial ones. Unlike the order of the equation, which is closely tied to the length of the statement, indicating linguistic labels, this method gives no general clues. In comparison to non-fuzzy models, the situation is more delicate, since no hypothesis (analogous to supporting linearity of the model) having any physical or at least numerical background can be formulated. It is simply a way of taking all the types of equation and verifying the extant to which each of them fits the data set provided.

To complete this presentation, we recall the model equation, using the remaining types of fuzzy-relational equation:

- for inf-s composition

$$Y_k = X_k \bullet \mathrm{R} \tag{5.10}$$

- for ad-joint fuzzy-relational equations:

$$Y_k = X_k \varphi R \tag{5.11}$$

From the data set $(X_k, Y_k), k = 1,2,\ldots,N$, the unknown fuzzy relation of the model is calculated:

$$R = \bigcup_{k=1}^{N} (X_k \beta Y_k) \tag{5.12}$$

and.

$$R = \bigcup_{k=1}^{N} (X_k \times Y_k) \tag{5.13}$$

It will be recalled that such fuzzy relations fulfill the corresponding equations using some assumptions. Thus, the direct use of these formulas is in line with the brute-force method.

Two algorithms show how a fuzzy relation can be estimated. The first refers to the probabilistic set already mentioned in the design of fuzzy controllers. Here we focus on the role of higher monitors in defining the precision of the derived model. The second uses a probabilistic layer to determine the fuzzy relation of the model.

5.3.3 Estimating the Fuzzy Relation of the Model by Probabilistic Sets

In many cases, the fuzzy sets creating the data set $(X_k, Y_k), k = 1,2,\ldots,N$, can be combined in (e.g. 'c') groups. Each cluster is an empirical representation of the probabilistic set, which thus reflects the structure of the data set. Their first two significant monitors are estimated from the partition matrix:

$$EX_i(x) = \frac{\sum_{c_i} X_k(x)}{n_i}$$

and

$$EY_i(y) = \frac{\sum_{c_i} Y_k(y)}{n_i}$$

The vagueness function is:

$$VX_i(x) = \frac{\sum_{c_i} [X_k(x) - EX_i(x)]^2}{n_i}$$

$$VY_i(y) = \frac{\sum_{c_i} [Y_k(y) - EY_i(y)]^2}{n_i}$$

$\mathbf{x} \in \mathbf{X}$, $\mathbf{y} \in \mathbf{Y}$

where the sum is taken over all the sets X_k, Y_k *belonging to the ith cluster;* n_i *is the* number of elements in this cluster.

Instead of requiring equation (5.8) to be solvable, we seek to solve:

$$EX_i \blacksquare \overline{R} = EY_i \qquad (5.14)$$

i = 1,2,..., c. Fuzzy relation R is estimated by the brute-force method:

$$\overline{R} = \bigcap_{k=1}^{N} (EX_i \varphi EY_i) \qquad (5.15)$$

Calculating the fuzzy relation R for the partition matrix, for c varying between 2 an N-1, can choose the number of clusters 'c'. The number of clusters is adjusted to minimize the index of quality of fit of the fuzzy model and the data set.

In a more general situation, the input-output data are probabilistic sets, given with their membership and vagueness function. Thus:

(EX_i, EY_i) , (VX_i, VY_i) , $i = 1,2,\ldots,c$

We may [3,7] regard any element of this data set, say a pair EX_i, VX_i, EY_i, VY_i (for brevity, we omit a subscript), as being distributed over intervals [a(x),b(x)] and [a(y),b(y)], so that their bounds are defined:

$$a(x) = EX(x)[1-VX(x)]$$
$$b(x) = EX(x)[1-EX(x)]VX(x)$$
$$a(y) = EY(y)[1-VY(y)]$$
$$b(y) = EY(y)[1-VY(y)]VY(y)$$

To obtain the bounds of the estimated fuzzy relation of the model, we restrict ourselves to sup-min composition in its equation. The upper bound of R is an α-composition:

$$R'(x,y) = a(x)\alpha \quad b(y) \tag{5.16}$$

while the lower bound is:

$$R'(x,y) = a(x)\alpha \quad b(y) = \begin{cases} a(y) & , \ if \quad b(x) \geq a(y) \geq \max\limits_{x=x_0} b(x_0) \\ 0 & , \ otherwise \end{cases} \tag{5.17}$$

This containment is satisfied:

$$R''(x,y) \leq R(x,y) \leq R'''(x,y) \tag{5.18}$$

where R is fuzzy relation of the model we are seeking. The closer R'' and R' are, the better the estimation. Thus, we define the vagueness of the estimation of the fuzzy relation R, VR, as:

$$VR(x,y) = R'(x,y) - R''(x,y)$$

For any subset of the data set, we can estimate the fuzzy relation of the model and express the vagueness of the estimate. This implies a procedure called the dynamic estimation algorithm, which is realized by (i) calculating the fuzzy relations R'' and R', and (ii) obtaining the relevant measures of vagueness and testing whether they are small enough to stop the algorithm. If the stopping criterion is not reached, the data set is modified or extended.

From the estimation procedure, we obtain two bounds of the fuzzy relation searched instead of just one as before, giving us much more information about the precision of the model derived. In further use of the model, we can take both bounds, leading us to an interval-valued fuzzy set.

Summarizing, in the probabilistic approach, the uncertainty is conveyed by the data set or stems from the discrepancy of the model itself, and is seen in the bounds of the computed estimates. One example where fuzzy sets are applied and we deal with non-fuzzy data is examined below [6].

The fuzzy-relational equation can be used effectively where we do not know which class of model (e.g. linear, polynomial etc.) is suitable. The data set consists of input-output pairs of non-fuzzy data, (1,1), (2,2), (3,1), (2,4), (4,1), (4,3), (4,4), (5,5). Taking the spaces **X** and **Y** as discrete, **X** = **Y** = (1,2,3,4,5), the optimal fuzzy relation is obtained by clustering the data (FUZZY ISODATA is used), yielding a relation with these entries:

1.00	0.22	0.00	0.08	0.00
0.04	0.09	0.00	0.08	0.00
1.00	0.09	0.00	0.08	0.00
0.04	0.09	0.00	0.08	0.00
0.05	0.11	0.00	0.08	1.00

The performance index optimized by the clustering method is the sum of the squared errors, namely

$$\sum_{i=1}^{8}(y_i - \bar{y}_i)^2$$

with y_i obtained by the center-of-gravity method. We study the performance of this method. When an input has a non-fuzzy value, the output is given by the appropriate row of the fuzzy relation R (the relation is normalized for each row). Its row is called a fuzzy profile of the model, denoted for a certain x_i by Fp(x_i). It implies a fuzzy set defined in **Y**. Thus, for each i, i = 1,2,..., 5, one has a corresponding fuzzy profile. The closer the fuzzy profile to any fuzzy set of output (or non-fuzzy value) in the data set, the better the fuzzy model obtained. The model is perfect if, for each input, the fuzzy profile is a singleton with a non-zero value of membership assigned to the output recorded in the data set. The fuzzy profiles for the model already constructed are displayed in Fig. 5.2. Good agreement with the data set is achieved. A linear model does not fit this data set.

We now show how knowledge of a probabilistic form of uncertainty can be handled in the framework of fuzzy models. We call this class of models, structured fuzzy models, since the original fuzzy models are constrained (structured) by the probabilistic ties.

5.3.4 Structured Fuzzy Models

We focus on the static fuzzy model with max-min composition:

$$Y = X \bullet R \tag{5.19}$$

For clarity, we assume that **X** and **Y** are finite spaces. Each pair of elements in the Cartesian product $\mathbf{X} \times \mathbf{Y}$ is characterized by the joint probability function $p(y_j, x_i), y_j \in \mathbf{Y}, x_i \in \mathbf{X}$. Evidently,

$$\sum_{i,j} p(y_j, x_i) = 1$$

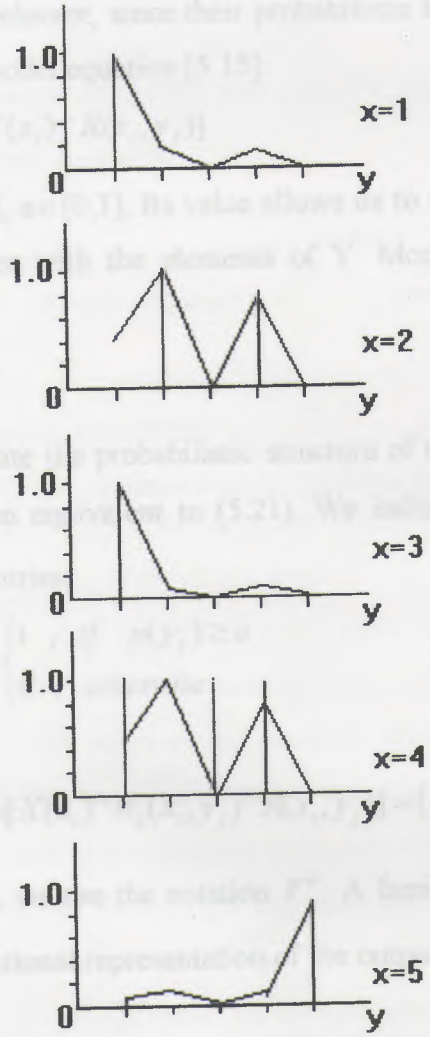


Fig. 5.2 Fuzzy profiles of the model

From this joint probability function, we obtain conditional probability functions $p(y_j|x_i)$:

$$p(y_j|x_i) = \frac{p(y_j, x_i)}{\sum_j p(y_j, x_i)} \tag{5.20}$$

expressing the strength of the transition (in the probabilistic sense) between the elements in spaces **X** and **Y**. All the transitions from a fixed x_i to u_j, j = 1,2,..., card(**Y**), give a sum of 1.0. This probabilistic layer is inserted into the model equation (5.20), in such a way that the indicated max-min composition is realized with respect only to those elements in **X** that have a sufficiently 'strong' probabilistic transition. The remaining elements are irrelevant, since their probabilistic transitions are negligible. In this light, we modify the model equation [5.15]:

$$Y(y_j) = \max_{x_i : p(y_i|x_i)} [X(x_i) \wedge R(x_i, y_j)] \tag{5.21}$$

'a' being a threshold level, a∈[0,1], Its value allows us to eliminate the elements of **X** that have insignificant ties with the elements of **Y**. More compactly, we write the previous formula:

$$Y = X \underset{p}{\bullet} Y \tag{5.22}$$

where the index 'p' indicate the probabilistic structure of the model equation. We can derive another formulation equivalent to (5.21). We indicate a Boolean relation B_a: **X**×**Y**→{0,1} with these entries:

$$B_a(x_i, y_j) = \begin{cases} 1 \ , & if \quad p(y_j) \geq a \\ 0 \ , & otherwise \end{cases} \tag{5.23}$$

Then we get:

$$Y(y_j) = \max_{x_i} [X(x_i) \wedge B_a(x_i, y_j) \wedge R(x_i, y_j)] = (X \bullet B_a \bullet R)(y_j)$$

To show the value of 'a', we use the notation Y^a. A family of fuzzy sets $\{Y^a\}_{a\in[0,1]}$ is taken as a joint fuzzy-relational representation of the output of the model. Two facts are evident:

- for a = 0, the influence of the probabilistic structure (layer) is disregarded,
- a < b implies $Y^a \subseteq Y^b$.

For some measure of uncertainty ascribed to Y^a. We estimate the fuzzy relation by using the fuzzy data $(X_i, Y_i), i = 1,2,\ldots,N$.

The data set is used in two ways: first to estimate the conditional-probability function; and second, in combination with the probabilistic characteristics, to calculate the fuzzy relation.

This computation uses a probabilistic strategy to eliminate some pairs in the data set that are biased and inconsistent with the probabilistic layer.

For estimating, we assume the above data set. Calculation of fuzzy relation R proceeds thus:

1. Start with fuzzy relation R having all entries equal to 1.0, $R(x_i, y_j) = 1.0$.
2. Put k = 1.
3. Calculate the α-composition of the kth pair from the data set, e.g. X_k and Y_k, and check simultaneously whether inequality

$$X_k \underset{p}{\bullet} X_k \alpha \ Y_k) = Y_k$$

is satisfied. If so, modify R, intersecting it with X_k α Y_k thus:

$$R = R \cap (X_k \alpha \ Y_k)$$

Otherwise, leave fuzzy relation R unchanged.

4. Increase k, k = k+1; if k is not greater than N, repeat 3; otherwise stop.

The output of the algorithm is simply the fuzzy relation of the model.

Let us explain the choice of threshold level 'a'. It is adjusted to meet a certain compromise. Too high an 'a' may cause only a few pairs of the data set to contribute to the computation; hence the fuzzy relation may be approximate. On the other hand, too low an 'a', e.g. about zero, may cause all the elements of the data set to contribute; thus the benefits of the probabilistic layer may be lost. It is reasonable to choose 'a' so as to minimize a performance index of the distance function between the fuzzy data and those fuzzy sets from the fuzzy model. Or we right use measure of the representative power of the identifying fuzzy sets (see in section 5.3.4).

We illustrate the above algorithm by an example of a Boolean data set with these input-output pairs:

5.3.5 Detecting the Structure of the Data Set

This section is devoted to algorithm for detecting the structure of the data set compiled for identification. The set is very useful, since only a few elements corrupted by noise strongly influence the estimation of the fuzzy relation. Our method compares the 'similarity' (or degree of equality) of the input and output fuzzy sets that are the elements of the data set. The degree of equality X_i and X_j is:

$$\gamma_{ij} = X_i = X_j = (X_j \subseteq X_j) \& (X_j \subseteq X_i) \tag{5.24}$$

where $X_i \subseteq X_j$ is the degree of containment of the fuzzy sets X_i and X_j. So, if $\gamma_{ij} = X_i = X_j$, $\xi_{ij} = Y_i = Y_j$, so that $\gamma_{ij} < \xi_{ij}$, we expect the data set consisting of two pairs (X_i, Y_i) and (X_j, Y_j) to be consistent. On the other hand, if $\gamma_{ij} > \xi_{ij}$, we can hardly solve the system of equations. We look at two situations:

First, γ_{ij} = 1.0 and ξ_{ij} = 0.0, which is the worst situation to solve, and, secondly, γ_{ij} = 0.0 and ξ_{ij} = 1.0, where the solvability is high.

Thus, we introduce this index:

$$\chi_{ij} = \gamma_{ij} \varphi \xi_{ij} \tag{5.25}$$

expressing the 'feasibility' of solving the equations. χ_{ij} is not equivalent in value to the solvability index, but is intermediately related to it. A modified version of the above equality index is of interest. We denote it by $(X_i = X_j)_{\max}$ and define it:

$$(X_i = X_j)_{\max} = \max_{x \in X} \{ [X_i(x) \varphi \ X_j(x)] \nparallel [X_j(x) \varphi \ X_i(x)] \} \tag{5.26}$$

The above index gives an optimistic measure of the equality of the two fuzzy sets.

So far, the feasibility χ_{ij} has involved only a pair of data, X_i and X_j. For a global view of the data set en bloc, we refer to some common hierarchical clustering algorithms. The objects grouped, i.e., the fuzzy sets, use the similarity of the pairs (X_i, Y_i) and (X_j, Y_j). The similarity matrix is $\chi = [\chi_{ij}]$, where:

$$\chi_{ij} = \chi_{ji} \quad , \quad \chi_{ii} = 1.0$$

To N clusters formed by single pairs from the data set, we apply an agglomerative procedure, so that the most similar clusters are merged. 1 reduces the number of clusters, and the procedure repeated until one cluster is left. The distance between clusters **X** and **Y** is:

$$d(X,Y) = \max(\rho_{ij}) \tag{5.27}$$

where a maximum is taken over all pairs X_i, Y_i and X_j, Y_j belonging to **X** and **Y**. ρ_{ij} stands for the distance function between two objects, say the ith and jth ones, ρ_{ij} = 1-χ_{ij}. The hierarchy generated by the distance specified above is known in clustering techniques as the complete linkage method [1]. The distance ρ_{ij} has a straightforward interpretation. When χ_{ij} specifies a degree of 'feasibility', the pair of fuzzy-relational equations allows us to solve the equation. ρ_{ij} Is the 'difficultly' in solving this pair of equations? Hierarchical clustering methods help to represent the data set; the above example allows us to study the data set, enabling us to compare these results with those achieved in setting probabilistic sets.

Example 5.1

The data set consists of N = 6 pairs of fuzzy sets with membership functions:

i	X_i	Y_i
1	[1.0 0.6 0.8 0.5]	[0.0 0.0 0.3 0.6]
2	[0.7 1.0 0.5 0.2]	[1.0 0.5 0.4 0.3]
3	[0.8 0.9 1.0 0.6]	[0.2 0.3 0.5 1.0]
4	[1.0 0.5 0.2 0.0]	[0.6 1.0 0.3 0.0]
5	[0.0 0.0 0.0 1.0]	[0.3 0.6 1.0 1.0]
6	[0.4 0.8 1.0 0.7]	[1.0 1.0 0.6 0.3]

The matrix of the model has these entries:

1.0	1.0	0.0	0.0	0.0	0.0
	1.0	0.2	0.0	1.0	1.0
		1.0	0.0	1.0	0.2
			1.0	0.0	0.0
				1.0	1.0
					1.0

From hierarchical clustering, we obtain the dendrogram in Fig. 5.3. At χ (calculated as a minimum of χ_{ij}, with i, j being the same objects in the same cluster) equal to 1.0, four clusters are seen: one containing three pairs {2,5,6} and the other three consisting of single pairs {3}, {1} and {4}.

As the number of clusters deceases, χ tends to zero; i.e., solvability of the system of equations (or its identification) is more difficult to achieve. The fuzzy relation, being an intersection of partial results (relations):

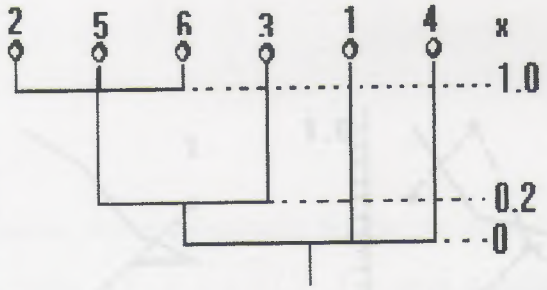


Fig. 5.3 Dendrogram from hierarchical clustering

$$\hat{R} = \bigcap_{i=1}^{6} \hat{R}_i$$

yields a poor performance, visible even without a formal specification of the index of solvability. We get this fuzzy relation:

0.0	0.0	0.3	0.3
0.0	0.0	0.3	0.3
0.0	0.0	0.3	0.3
0.0	0.0	0.3	0.3

Thus, the fuzzy set $X_i \bullet \hat{R}$ differs significantly from the original Y_i: no one $X_i \bullet \hat{R}$ has a membership function greater than 0.3. The results of max-min composition are shown in Fig. 5.4. Taking the fuzzy relation equal to the intersection of $\hat{R}_2, \hat{R}_5$, and $\hat{R}_6$, say $\hat{R}_{256}$, based on the data forming one cluster in the previous dendrogram, we get:

1.0	0.5	0.4	0.3
1.0	0.5	0.4	0.3
1.0	1.0	0.4	0.3
0.3	0.6	0.6	0.3

For comparison, the resulting fuzzy sets $X_i \bullet \hat{R}_{256}$ are also summarized in Fig. 5.4. The solvability index of the fuzzy system equations is higher now, but does not exceed the specified value of the index χ. Recalling the results of the probabilistic sets obtained by iterative clustering (in [5.3.6], ISODATA was used), we get this fuzzy relation:

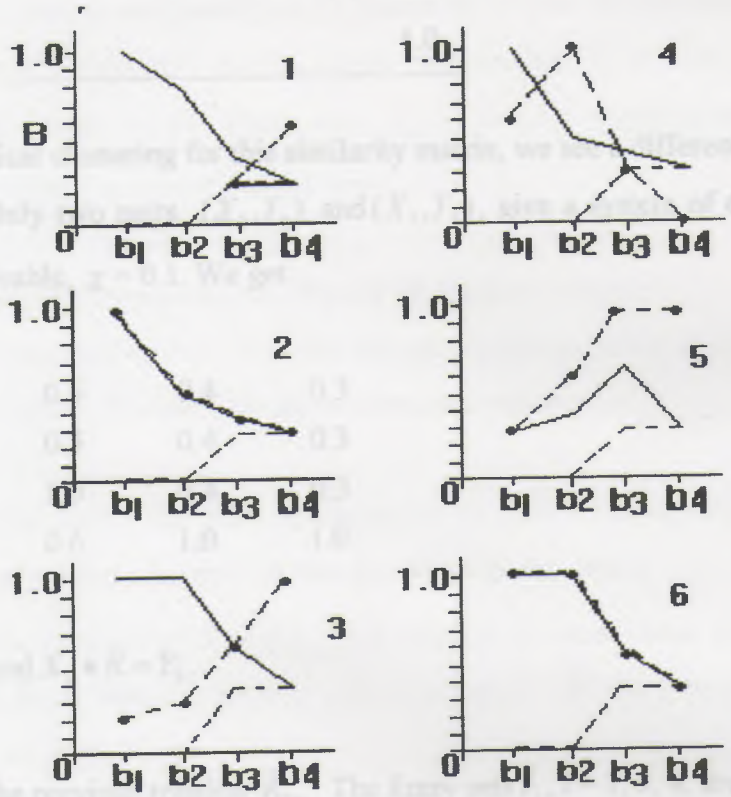


Fig. 5.4 Membership functions of the output of the model produced by various fuzzy relations

0.53	0.50	0.33	0.30
0.53	0.50	0.33	0.30
0.60	0.65	0.33	0.30
0.30	0.60	0.55	1.00

In this case, the data set contains three clusters, namely {1,2,4}, {5} and {3,6}. There are no significant differences between the entries of this fuzzy relation and that computed elsewhere, namely $\hat{R}_{256}$. Taking now χ_{ij}, defined by means of $(X_i = X_j)_{\max} \varphi(Y_i = Y_j)$, we obtain:

1.0	0.0	0.0	0.0	0.0	0.0
	1.0	0.2	0.0	1.0	0.4
		1.0	0.0	0.2	0.2
			1.0	0.0	0.0
				1.0	0.3
					1.0

Using hierarchical clustering for this similarity matrix, we see a difference from the first dendrogram. Only two pairs, (X_2, Y_2) and (X_5, Y_5), give a system of equations that is completely solvable, $\chi = 0.1$. We get:

1.0	0.5	0.4	0.3
1.0	0.5	0.4	0.3
1.0	1.0	0.4	0.3
0.3	0.6	1.0	1.0

$X_2 \bullet \hat{R} = Y_2$, and $X_5 \bullet \hat{R} = Y_5$.

We focus on the previous relation, $\hat{R}_{256}$. The fuzzy sets $\hat{Y}_i$, i = 1, 3, 4, are greater than the original ones. This is not surprising, because fuzzy sets X_2, X_5, X_6 do not 'cover' the entire space X. Thus, fuzzy relation $\hat{R}_{256}$ does not map the data set very well. However, we add the next pair, e.g. (X_3, Y_3) (see the dendrogram), and fuzzy relation $\hat{R}_{2563}$ gives a low $\chi (\chi = 0.2)$; the fuzzy relation is:

0.2	0.3	0.4	0.3
0.2	0.3	0.4	0.3
0.2	0.3	0.4	0.3
0.2	0.3	0.4	0.3

In this analysis, we ask for the number of pairs of fuzzy data that come into the computation of the fuzzy relation. In other words, what threshold level in the dendrogram indicates the fuzzy sets that give the most reliable and consistent subset of the entire data set? This choice is not a trivial task. On the one hand, too many inconsistent fuzzy data in computing the fuzzy relation can give meaningless results, such as a fuzzy relation with nearly all its entries set to zero. On the other hand, too few fuzzy data prevent discovery of the entries relationship. It is vital to evaluate the 'representative' power of the fuzzy sets $X_1, X_2, \ldots, X_N$ contributing to the fuzzy relation. This problem is studied below.

5.3.6 Measuring the Representative Power of the Fuzzy Data

A concept enabling us to measure the representative power of a data set is based on a simple statement. Consider the fuzzy-relational equation:

$$X \bullet R = Y$$

with X,Y, R specified in the same spaces as before. From max-min composition, it is evident that only those elements of the fuzzy set X for which $X(x_j) \geq Y(y_k)$ holds contribute to the determination of the fuzzy relation. In other cases, the corresponding element of the fuzzy relation, namely (j,k), is equal to 1.0. We introduce an auxiliary vector defined in space X:

$$v = [v_1 \quad v_2 \quad v_n] \tag{5.28}$$

where each entry is:

$$v_j = card\{y_k \mid X(x_j) \geq Y(y_k)\} \quad , \quad j = 1,2,\ldots,n$$

This gives the number of events where the membership function X at point x_j exceeds or is equal to the membership function Y at point y_k. Replacing v by a probability vector those results from simple normalization of v gives:

$$p = [p_1 \quad p_2 \quad \cdots \quad p_n]$$

with

$$p_j = \frac{v_j}{\sum_{l=1}^{n} v_l} \tag{5.29}$$

If the sum of v_js is zero, p_j is zero as well. Vector p shows how well fuzzy relation R is determined and its best-estimated rows. The higher p_j is, the better the jth row of R is determined. If X is given as follows: $X(x_j) = \delta_{j,j_0}$, the corresponding probability is:

[0.0 ... 0.0 1.0 0.0 ... 0.0]

with 1.0 set at the $j_0 th$ position. The $j_0 th$ row of R is well estimated. But, taking X so that all v_js are zero (vector of probabilities p = 0), we find that the fuzzy relation cannot be estimated. In an intermediate situation where X is 'unknown' (i.e., its membership function is 1.0 over the whole universe of discourse), for all x_j the value v_j = m and, in consequence, p = [1/n 1/n ... 1/n]. So all the rows of the fuzzy relation equal 1/n.

For a system of fuzzy-relational equations, we can assign to each pair of data a vector of probability p_i and perform a global evaluation:

$$p = \bigvee_{i=1}^{N} p_i \tag{5.30}$$

Notice that p_i allows us to eliminate some data in the preliminary analysis: all the fuzzy data whose vector of probability is 0 can be excluded. The highest p is attained when the elements of the data set are carefully chosen. Here are some general hints:

- if fuzzy sets X_i, i = 1,2,..., n (i.e., satisfy the conditions:

(i) max $X_i(x_j) = X_i(x_i)$, $i = 1,2,...n$

and

(ii) $\underset{i=1,2,...,n}{\forall} \underset{k=1,2,...,m}{\forall} \underset{j=i}{\forall} \{y \mid X(x) \geq Y(y)\} = \succ$

then p = [1.0 1.0 ... 1.0] = 1.

Loosely speaking, fuzzy sets X_i should be maximum at just one point of the universe of discount (x_i), and should be 'sharp' enough (condition (ii)). The value of p for our data set tells us how suitable the fuzzy are sets used to determine the fuzzy relation. The closer p is to 1, the better the data set is for relation estimation; but only potentially, since all the pairs are taken separately. The whole picture is seen when we

combine this approach with the overall analysis given by the dendrogram. This illuminates the data set from two different, competitive, points of view, namely:

- ability to solve the system of equations,
- ability to represent the fuzzy relation.

These are contradictory. Moving from the top of the dendrogram to the bottom, it is more difficult to satisfy the first requirement. In the opposite direction, the same holds for the second need. We return to our numerical example:

$$p_1 = [0.27 \quad 0.27 \quad 0.26 \quad 0.20]$$
$$p_2 = [0.33 \quad 0.44 \quad 0.23 \quad 0.00]$$
$$p_3 = [0.23 \quad 0.23 \quad 0.31 \quad 0.23]$$
$$p_4 = [0.67 \quad 0.33 \quad 0.00 \quad 0.00]$$
$$p_5 = [0.00 \quad 0.00 \quad 0.00 \quad 1.00]$$
$$p_6 = [0.11 \quad 0.22 \quad 0.44 \quad 0.23]$$

If we restrict ourselves to a compact cluster of the elements in the data set numbered 2, 5 and 6, for which $\chi = 0.1$, the overall vector p has these entries:

$$p = p_2 \vee p_5 \vee p_6 = [0.33 \quad 0.44 \quad 0.44 \quad 1.00]$$

Adding the pair (X_3, Y_3) resulting from the dendrogram does not change vector p. At the same time χ decreases drastically. The ability of the whole data set to estimate the fuzzy relation is:

$$p' = [0.67 \quad 0.44 \quad 0.44 \quad 1.00]$$

This indicate that restricting the first subset of the data set to only three pairs does not significantly decreases its ability to determine the fuzzy relation. A change in p' compared to p is observed for only one element of space X. The method presented is suitable for analyzing the structure of the fuzzy data collected to determine the fuzzy relation. An index of the ability of a subset of data to estimate the fuzzy relation has been introduced. Nevertheless, the choice of threshold level in the dendrogram is open, and is up to the user. In every situation, some compromise is needed.

We discuss now a method of solving the system equations by introducing an additional space.

5.3.7 Fuzzy Models with Additional Variables

The essence of this estimation algorithm is an additional space in which the 'decoupling' fuzzy sets are defined. A loose analogy is found in model building in a random environment. An output variable, say y, is tied to the input variable x by a function dependence, y = f(x). If f is a linear function (it is usually assumed as a preliminary from of a model, if no clear indications are available), two parameters must be estimated, 'a' and 'b', so that y = ax + b. This is realized, for example, by minimizing the sum of the squared errors,

$$\sum_{i=1}^{N}(ax_i + b - y_i)$$

with pairs (x_i, y_i) forming a data set (corresponding to the pairs of fuzzy data X_i and Y_i discussed here). Afterwards, the model is tested against the data set, using some powerful statistical-inference methods (e.g. the F-test). If the model is statically inconsistent, it may be inadequate. This is detected by a statistical test. In this situation, it reasonable to try a more complex form of model, not simply linear, but incorporating some terms of higher order. Thus, we are forced to introduce, for instance, a model of the form y = c x^2 + ax + b. Now, a new variable w = x^2 has to added. The model is extended to one having two input variables, in which one is created in an artificial manner. Then the model is tested against the data set consisting of triples $(w_i, x_i, y_i) = (x_i^2, x_i, y_i)$. If that an extended model is derived.

Along these lines, we reformulate the estimation of the fuzzy relation. To do this, we write:

$$(X_i \times C_i) \bullet G = Y_i \tag{5.31}$$

where $C_1, C_2, \ldots, C_N$ are fuzzy sets defined in the auxiliary space C, and fuzzy relation G is expressed in $X \times C \times Y$. Fuzzy sets C_i are chosen so that the particular equations are decoupled (to be clarified later). The most 'difficult' situation (in the sense of satisfying the solvability of the system of equations) is:

$$(X_i = X_j) > (Y_i = Y_j)$$

We can Improve this by introducing normal fuzzy sets C_i and C_j and calling the decoupling fuzzy sets, for which the inequality:

$$(X_i \times C_i = X_j \times C_j) < (Y_i = Y_j) \tag{5.32}$$

is satisfied. More formally, we state:

Proposition 5.2

For all fuzzy sets X_i, X_j, C_i, C_j, such that $C_i \cap C_j = \phi$, we have:

$$(X_i \times C_i = X_j \times C_j) \leq (Y_i = Y_j)$$

To make the system of equations solvable (assuming that each equation of this set of equations does have a solution), it is sufficient to add a new space. Hence, the purpose of the extra fuzzy sets is permanent: thus, they can be found for any system of equations. With this modification, any system of equations (assuming each equation treated separately has a solution) is solvable. The only question is how to choose the decoupling fuzzy sets for the given system of equations. The simplest and almost self-evident way is to treat the space $C = (c_1, c_2, \ldots, c_N)$ and the fuzzy sets $C_1, C_2, \ldots, C_N$ as singletons:

$$C_i(c_j) = \delta_{ij} \quad , \quad i, j = 1,2,\ldots,N \tag{5.33}$$

But this choice is not unique. Now we offer a useful method of finding the biggest decoupling fuzzy sets for a system of equations. They have an interesting property. They express how strongly the membership functions in the extra space may overlap to ensure that the system of equations is still solvable. If an original system of equations is solvable, the biggest decoupling fuzzy sets have membership functions equal to 1.0 over the entire space C. Consider some clusters of data detected by hierarchical clustering. Each cluster contains several pairs (X_i, Y_i) from the data set. We seek the decoupling fuzzy sets $C_1, C_2, \ldots, C_P : C \rightarrow [0,1]$ with $C = \{c_1, c_2, \ldots, c_N\}$. 'P' is the number of clusters. Thus, the data of a format $(X_{i1}, Y_{i1}, \ldots, X_{iP}, Y_{iP}), P = card(X_r)$, r = 1, 2, ..., P, coming from the rth cluster, are extended into triples, $(X_{i1}, C_P, Y_{i1}), \ldots, (X_{iP}, C_P, Y_{iP})$.

The membership functions of the fuzzy sets C_P are computed:

$$\begin{aligned} C_P(c_P) &= 1.0 \\ C_P(c_r) &= \wedge(Y_i = Y_j) \end{aligned} \tag{5.34}$$

where a *minimum* is taken for all the pairs so that:

$$(X_i, Y_i) \in X_r \quad , \quad (X_j, Y_j) \in X_P \quad (X_i = X_j)_{\max} \geq (Y_i = Y_j)$$

We adopt here the notation

$$\underset{0}{\wedge}(Y_i = Y_j) = 1.0$$

The results of this computation are shown in Fig. 5.5. This is an

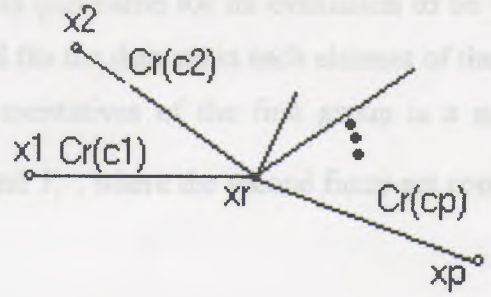


Fig.5.5 Decoupling fuzzy sets

undirected graph with nodes formed by the clusters detected, and edges showing the strength of the relations between the clusters. Returning to our examples and splitting the data set into four clusters, $X_1 = \{(X_1, Y_1)\}$, $X_2 = \{(X_2, Y_2), (X_5, Y_5), (X_6, Y_6)\}$, $X_3 = \{(X_3, Y_3)\}$, $X_4 = \{(X_4, Y_4)\}$, we have membership functions of the decoupling fuzzy sets equal to:

$$C_1 = [1.0 \quad 0.0 \quad 0.0 \quad 0.0]$$
$$C_2 = [0.0 \quad 1.0 \quad 0.2 \quad 0.0]$$
$$C_3 = [0.0 \quad 0.2 \quad 1.0 \quad 0.0]$$
$$C_4 = [0.0 \quad 0.0 \quad 0.0 \quad 1.0]$$

When every cluster consists of exactly one pair, $X_i = \{(X_i, Y_i)\}$, the biggest decoupling fuzzy sets are:

$$C_1 = [1.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0]$$
$$C_2 = [0.0 \quad 1.0 \quad 0.2 \quad 0.0 \quad 1.0 \quad 0.4]$$
$$C_3 = [0.0 \quad 0.2 \quad 1.0 \quad 0.0 \quad 0.2 \quad 0.2]$$
$$C_4 = [0.0 \quad 0.0 \quad 0.0 \quad 1.0 \quad 1.0 \quad 0.0]$$
$$C_5 = [0.0 \quad 1.0 \quad 0.2 \quad 1.0 \quad 1.0 \quad 0.3]$$
$$C_6 = [0.0 \quad 0.4 \quad 0.2 \quad 0.0 \quad 0.3 \quad 1.0]$$

5.3.8 Evaluation of the Fuzzy Model

Is our fuzzy model suitable? We distinguish between two situations, the first referring to a general mode of model testing and the second concerning evaluation method for particular applications, such as prediction and control. Generally, evaluation is more synthetic, giving a yes-no response for the global model. If the special task of

the model is solved, it is preferable for its evaluation to be point-wise. Thus, we can judge whether the model fits the data set in each element of the universe of discourse Y.

One of the representatives of the first group is a sum of distance functions between fuzzy sets Y_i and Y_i', where the second fuzzy set comes from the fuzzy model, $Y_i' = X_i \blacksquare \mathrm{R}$:

$$D = \sum_{i=1}^{N} d(Y_i, Y_i') \tag{5.35}$$

where d(...) is the distance function, e.g., Hamming, Education or, more generally, Minkowski. The acceptance or rejection of the model is now straightforward:

accept the model if $\mathrm{D} < \mathrm{D}_{crit}$

where D_{crit} is the critical value of the sum of the distance. Thus, if D is smaller, the model is acceptable. Shortcomings referring to this approach also are evident:

- we cannot express how the model fits the data set,
- the choice of critical value (D_{crit}) is open, with no reasonable underlying selection criterion.

The second situation is illustrated below.

5.3.8.1 Model evaluation by the fuzzy-measure approach

First, all the output fuzzy sets and the corresponding fuzzy sets are point-wise compared. Thus, for each Y_i and Y_i', we get a fuzzy set of their equality $\Gamma_i : Y - [0,1]$, where

$$\Gamma_i = Y_i = Y_i' \tag{5.36}$$

i.e.,

$$\Gamma_i(y_j) = [Y_i(y_j) \varphi Y_i'(y_j)] \not\vdash [Y_i'(y_j) = Y_i(y_j)]$$

For a global evaluation of the equality set describing the model, an aggregation is needed. It is not unique, however, and these forms of aggregation are of interest:

- pessimistic:

$$\Gamma = \bigcap_{i=1}^{N} \Gamma_i$$

- optimistic:

$$\Gamma = \bigcup_{i=1}^{N} \Gamma_i$$

- average:

$$\Gamma = Av(\Gamma_i) \quad , \quad \Gamma(y_j) = \frac{\sum_{i=1}^{N} \Gamma_i(y_j)}{N}$$

If $\Gamma = 1$ (all elements of this vector are equal to 1), we assume that our model completely fits the data set. Conversely, if Γ tends to 0, we say that the model is weak.

The quality of the model using fuzzy set Γ is expressed by a fuzzy measure. For clearly, we consider the output space Y and say that the fuzzy model evaluated in terms of satisfaction of the property [4] is:

'the space Y well mapped (well represented) by the fuzzy model under evaluation' (5.37)

Thus, for any input X, the output of the model, and the fuzzy set describing the output of the system, are indistinguishable, or at least close to each other, especially for the fuzzy data used in constructing the fuzzy model (X_i, Y_i). For two fuzzy models with respective fuzzy sets Γ and Γ', we have two situations:

(i) $\Gamma \subset \Gamma'$; i.e. all the coordinates of Γ' are grater than or equal to the corresponding coordinates of Γ. Here, the model with Γ works worse than that with Γ' (it is less precise).

(ii) Γ and Γ' are not comparable.

Situation (ii) is more frequent than situation (i). This fact, as well as the point-wise construction of Γ, suggests a partial evaluation of the model for each coordinate, i.e. the fulfillment of a 'local' property:

'the ith coordinate of Y is well mapped by the fuzzy model' (5.38)

On the Gestalt principle, the global property (5.37) cannot be deduced from a simple, perhaps linear, aggregation of partial evaluations of the model using (5.38). This leads us to a fuzzy measure as a plausible tool for global evaluation of the fuzzy model. As

we shall see, the structure of the fuzzy integral enables us to link the quality of the model and the controllability and predictability of the system.

Fuzzy measure g_λ is a set function defined in the Borel field B as monotonic with boundary $g_\lambda(\phi)=0$ and $g_\lambda(Y)=1$. Fuzzy measure $g_\lambda(.)$ 'measures' the quality of the model for any subset of Y. If the known number of elements of Y with concrete levels of property (5.38) increases, our ability to judge the model as a whole increases. This is reflected by the monotonicity of the fuzzy measure. Without a point of Y, we can make no judgment. If we know our model quality in only one point of Y, say y_j, we know it from $g_\lambda(\{y_j\})$, which is identical with our previous evaluation through Γ_j. Here, we can make a partial judgment about the model, but it may be marginal. Considering only a single element y_j, significant overestimation of the quality of the model may occur if $\Gamma(y_j)=1$. Conversely, if $\Gamma(y_j)=0$, we underestimate its performance.

From fuzzy set Γ, the λ-fuzzy measure is determined numerically. From the λ-rule, the fuzzy measure is:

$$g_\lambda(Y')=\frac{\left\{\prod_{y_i\in Y'}(1+\lambda\Gamma(y_i))-1\right\}}{\lambda}\quad,\quad \lambda\in(-1,\infty)$$

5.3.8.2 Evaluation of the fuzzy model by using the introduced confidence levels

Now we give some analogies to compare the methods used to evaluate the statistical (probabilistic) models. We discuss a model of first order:

X ■ R = Y

assuming that the identifying fuzzy data (X_i,Y_i) are available. The fuzzy relation of the model (omitting its origin) yields a fuzzy set Y_i' for X_i treated as the input of the model. To summaries the membership functions of Y_i and Y_i' for all i = 1, 2, ..., N in a fixed element of the universe of discourse, say y_j, we construct an empirical distribution function of the equality index:

$$F_j(w) = \frac{card\left\{i \mid (Y_i(y_j) = Y_i^{'}(y_j)) < w\right\}}{\mathrm{N}}$$

$w \in [0,1]$

The interpretation of $F_j(w)$ is straightforward: it articulates the probability that the equality index is larger than 'w'. In the extreme:

- if the model fits perfectly the data set at the specified element of the universe of discourse y_j, all the values of the equality index $Y_i(y_j) = Y_i^{'}(y_j)$ are set to 1.0, and therefore:

$$F_j(w) = \begin{cases} 0 & , \quad if \quad w < 1 \\ 1 & , \quad otherwise \end{cases}$$

so the distribution function has one jump of height at 1.0, located at w = 1.0,

- if the model gives results different from the data in the set, the distribution function is 1.0 for all arguments, with precisely one step at w = 0.0.

We can usually deal with intermediate situations, where the distribution function has a step-like function, by steps of different heights scattered over the whole unit interval. The better model has a distribution function with non-zero values moved towards higher values of the argument.

The equality index γ has this probability:

$$\alpha = \mathrm{P}\left\{w \mid (Y_i(y_j) = Y_i^{'}(y_j)) \geq \gamma\right\}$$

(we skip index 'j' in the equality index γ, as well as α, assuming entire investigation is performed for each j, j = 1, 2, …, card(Y)).

Probability α expresses the fraction of the grades of membership function for which the equality index exceeds γ. From the obvious relationship:

$$\mathrm{P}\left\{w \mid (Y_i(y_j) = Y_i^{'}(y_j)) \geq \gamma\right\} = 1 - \mathrm{P}\left\{w \mid (Y_i(y_j) = Y_i^{'}(y_j)) < \gamma\right\} = 1 - F_j(\gamma)$$

we get:

$$\alpha = 1 - F_j(\gamma)$$

For each fixed α, we can determine the corresponding γ. More precisely, we take a maximal γ, denoting by $\Gamma = \{\gamma \mid 1 - \alpha = F_j(\gamma)\}$ the solution γ_*, taken as

$$\gamma_* = \max_{\gamma \in \Gamma} \arg \gamma$$

Conversely, from the equality index γ, we get the probability. Thus, we see a function dependence between γ and α, $\gamma = f(\alpha)$, giving an interesting insight into the probability of occurrence of the specified values of the equality index. Further analysis of this expression allows us to discover some properties of the function $\gamma = f(\alpha)$. We rewrite it as $F_j(w) = 1 - \alpha$. If α increases, $1 - \alpha$ decreases; since the distribution function is a non-decreasing function, we get lower values of the equality index. This phenomenon can be expected: if we want to achieve high probability, it occurs at lower γ. retransforming the grade of membership with the given γ, we get broader and broader equality intervals. The equality index has its analogue in a confidence interval found in statistics, especially in parameter estimation. For a higher value of confidence, say 0.01, as compared to a particular standard of 0.05, the intervals are quite wide.

Computing thus for all the elements of Y, we get the corresponding distribution functions of the equality indices. From the membership function of the output fuzzy set, we get the equality intervals for the prespecified γ or α.

We consider two extremes of behavior of the above fuzzy model. In the first, the model fits completely the data set and the distribution function of the equality indices possesses one jump of 1.0 at w = 1.0. This implies that, for all $\alpha \in [1,0]$, $\gamma = \gamma(\alpha) =$ const = 1.0, and the corresponding equality interval decreases to a point. Thus, we get a genuine fuzzy set, not an interval-valued one, which shows that the model is extremely precise.

In the second case, the fuzzy model produces a completely different fuzzy set. Now, the distribution function has, as before, one jump of 1.0, but this jump is put at w = 0.0. By straightforward computation, we can verify $\gamma = \gamma(\alpha) =$ const = 0.0, which gives an equality interval of [0,1]. Thus, the model is irrelevant and the resulting fuzzy set conveys no useful information.

An example illustrates the use of the confidence intervals of the fuzzy model. The data set has already been discussed. The model has the form $Y = X \bullet R$. Three ways estimating the relation of the model have been used. The first (a) is a brute-force one; the second (b) is based on probabilistic sets; the third (c) considers a subset of the data set that has high consistency and is extracted by hierarchical clustering. The plots are shown in Figs. 5.6(a), (b) and (c). The brute-force method produces

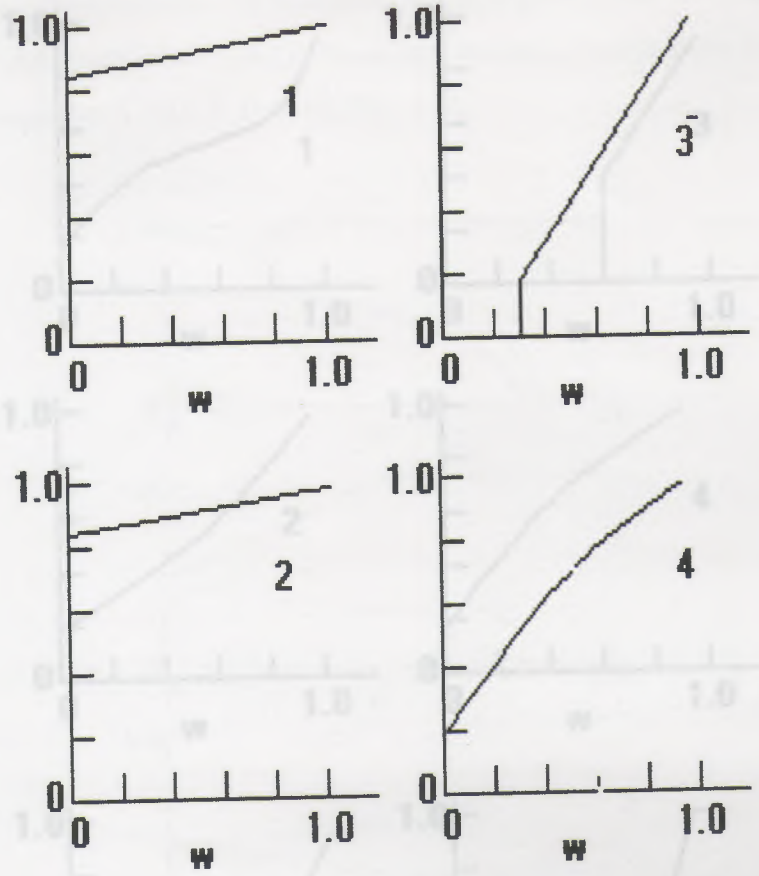
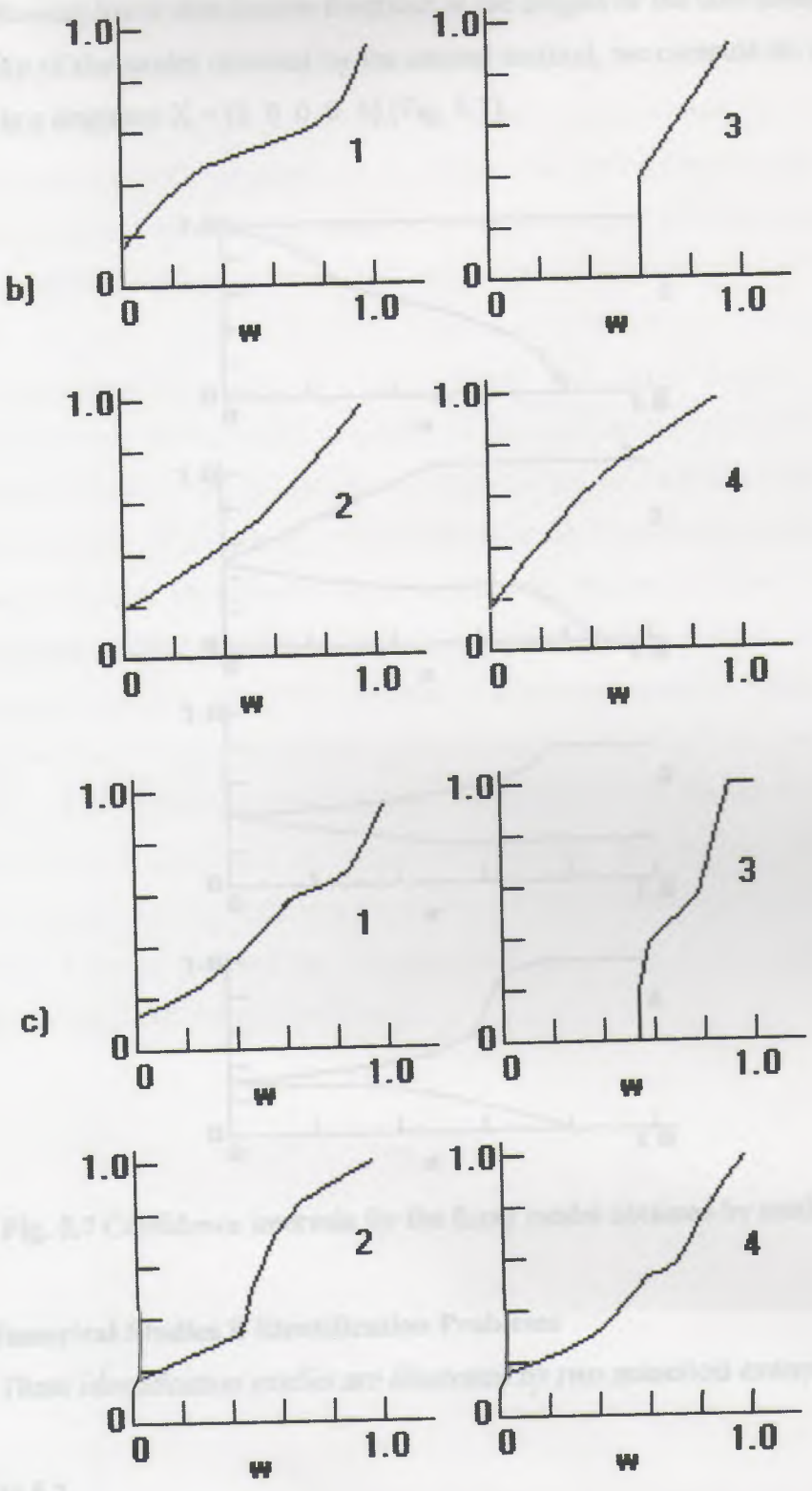


Fig. 5.6(a) Confidence intervals for the fuzzy models versus α



Figs. 5.6(b) and (c) Confidence intervals for the fuzzy models versus α

poor results, especially at the first and second elements of the universe of discourse Y (the distribution functions at w = 0.0 are near to 1.0). The other two methods give better

results, allowing lower distribution functions at the origins of the coordinates. To show the equality of the model obtained by the second method, we compute its output when the input is a singleton X = [1 0 0 0 0] (Fig. 5.7).

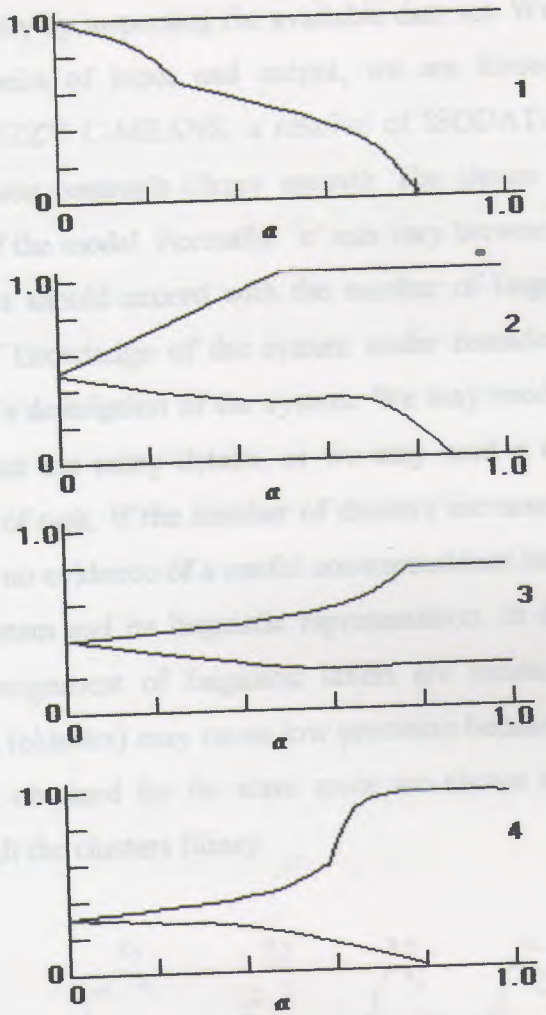


Fig. 5.7 Confidence intervals for the fuzzy model obtained by method (b)

5.3.9 Numerical Studies if Identification Problems

These identification studies are illustrated by two numerical examples:

Example 5.2

A gas-furnace data set is a standard test for identification and estimation. In fuzzy sets, Tong has used it. However, his approach, heuristic in nature, is completely different from that presented here. The data set consists of n = 296 parts of input-output

observations, where the input is the rate of gas flow into the furnace and the output is the concentration of CO_2 in the outlet gases. The sampling intervals are 9 s.

Since the data are non-fuzzy, and we intend to build a fuzzy model describing the relations between the linguistic (fuzzy) labels, we start by looking for them. This can be done conveniently by inspecting the available data set. Without extra knowledge about the collected pairs of input and output, we are forced to use a clustering technique. We use FUZZY C-MEANS, a relative of ISODATA, which produces 'c' clusters and detects their centroids (fuzzy means). The choice of number of clusters depends on the user of the model. Formally. 'c' can vary between 2 and N-1. However, the number of clusters should accord with the number of linguistic labels, reflecting naturally the level of knowledge of the system under consideration, or the level of generality in the user's description of the system. We may need a concise and east-to-use description without too many details, or we may need a detailed description for solving special types of task. If the number of clusters increases, a higher precision is obtained, but there is no evidence of a useful correspondence between the formal fuzzy description of the system and its linguistic representation. In the extreme, c = N, the establishment and assignment of linguistic labels are meaningless. But too low a number of categories (clusters) may cause low precision because the model obtained is too general. Clusters obtained for the state space are shown in Fig. 5.8 (putting five clusters). We shall call the clusters (fuzzy

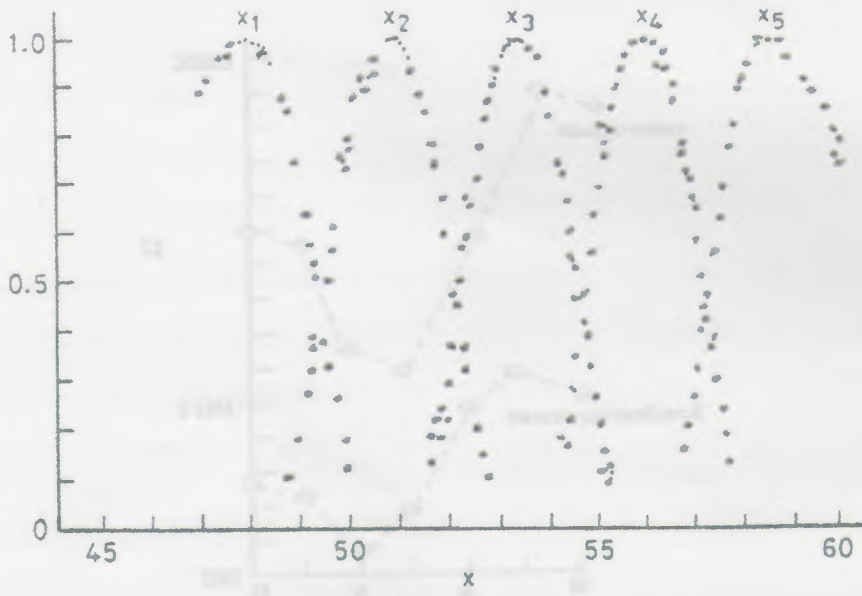


Fig. 5.8 Membership functions of the linguistic labels defined in the state space

sets) reference fuzzy sets. Thus, for space X, we distinguish $X_1, X_2, \ldots, X_{cX}$, while the fuzzy sets in the control space U are denoted by $U_1, U_2, \ldots, U_{cU}$. The form of the model is given by this fuzzy-relational equation:

$$X_{k+1} = U_{k-\tau} \blacksquare X_k \blacksquare \mathrm{R}$$

$$\tau = 0, 1, 2$$

The composition operator in the model equation is fixed as max-min or max-product. $U_{k-\tau}, X_k, X_{k+1}$ are fuzzy sets defined in U and X, respectively. By U and X we mean a family of he reference fuzzy sets:

$$\mathrm{U} = \{U_1, U_2, \ldots, U_{cU}\} \quad , \quad \mathrm{X} = \{X_1, X_2, \ldots, X_{cX}\}$$

All the elements of the data set are transformed into fuzzy sets by means of our reference fuzzy sets. Based on the theory of fuzzy-relational equations, the fuzzy relation of the model is computed. Our performance index is the sum of the squared distances between the fuzzy sets obtained from the model, denoted by X'_{k+1}, and those from the data set X_{k+1}:

$$Q = \sum_{k=1+\tau}^{N-1} \sum_{i=1}^{c_X} (X_{k+1}(i) - X'_{k+1})^2$$

The performance indices for different delay time's τ and for the two composition operators are shown in Fig.5.9. Hence, the max-product

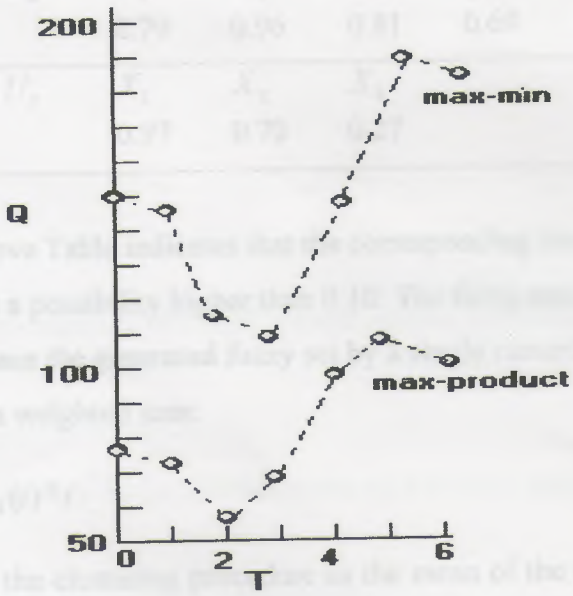


Fig. 5.9 Q versus τ for max-min and max-product compositions

composition operator is preferable with τ set to 2. Since there are five clusters in spaces X and U, the fuzzy relation consists of 5^3 elements. We can give it a more compact form. Each entry of R can be viewed as an implication statement with its own possibility measure. From this relation, we derive this linguistic description of the system:

if

the state of the system at the kth time instant is X_i and the control applied is U_j

then

the state at the $(k+1)$th time instant is X_l

The rules are given in decreasing order of the possibility measure, or they may be stored in this matrix form:

	X_1	X_2	X_3	X_4	X_5
U_1	--	--	X_5 0.92	X_4 0.76	X_5 0.98
U_2	--	X_3 0.70	X_3 0.82	X_4 0.92	X_5 0.95
U_3	X_4 0.49	X_2 0.89	X_3 0.99	X_4 0.98	X_5 0.99
U_4	X_1 0.79	X_2 0.96	X_3 0.81	X_4 0.69	X_5 0.10
U_5	X_1 0.97	X_2 0.70	X_2 0.27	--	--

The sign - - in the above Table indicates that the corresponding state and control produce no state with a possibility higher than 0.10. The fuzzy model can be used numerically. We replace the generated fuzzy set by a single numerical quantity by taking, for example, a weighted sum:

$$x_{k+1} = \sum_{i=1}^{c_X} X_{k+1}(i) * i$$

Where *i is given in the clustering procedure as the mean of the ith cluster. Measuring the quality of the fuzzy model by the sum of the squared errors:

$$\sum_{k=\tau_{top}}^{N-1}(X_{k+1} - *_{k+1})^2$$

(Where τ_{opt} is the optimized delay) or an average of this index, say Q_{av} = Q/(N-τ_{opt} -2), we get, for 5,7 and 9 clusters in X and U:

$$c_X = c_U = 5 \quad , \quad Q_{av} = 0.776$$

$$c_X = c_U = 7 \quad , \quad Q_{av} = 0.478$$

$$c_X = c_U = 9 \quad , \quad Q_{av} = 0.320$$

The model becomes more precise as the number of reference fuzzy sets increases. Bearing in mind our remark above about the number of linguistic labels, we have a contradiction, stressed in Zadeh's principle of incompatibility. The results of the model for five and nine reference fuzzy sets are shown in Fig. 5.10.

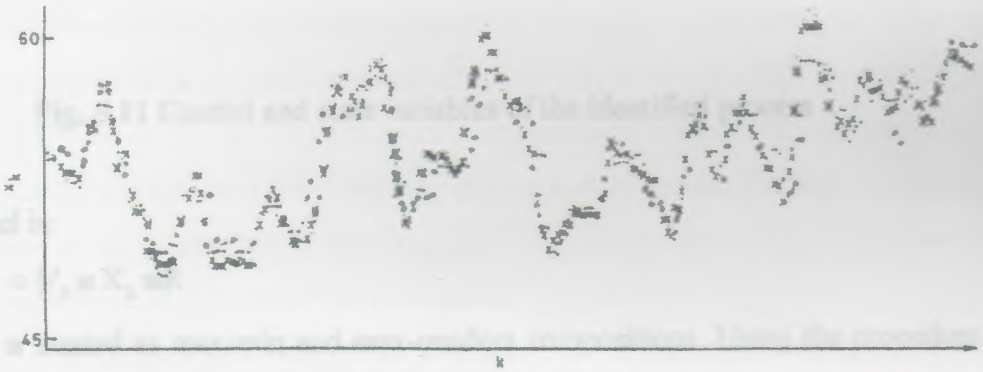


Fig. 5.10 performance of the fuzzy model with no fuzzy representation

Example 5.3

Our second example is the identification of a system described by a difference nonlinear equation:

$$x_{k+1} = 0.8x_{k-1} + 0.1x_{k-2} + 0.3x_{k-1}u_{k-1} + 0.8u_{k-1} + z_k$$

Where z_k stands for the Gaussian disturbance with zero mean value and standard deviation $\sigma_z = 0.05$. The data set for identification is given in Fig. 5.11 and consists of 50 pairs of input (u_k, x_k). The fuzzy

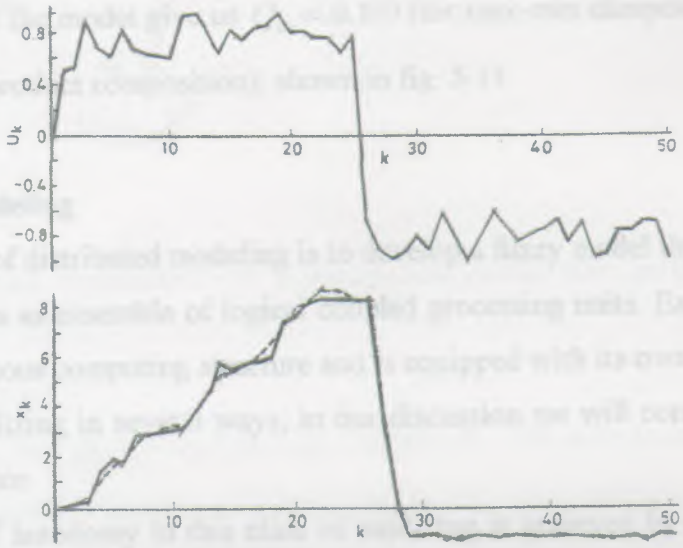


Fig. 5.11 Control and state variables of the identified process

Model is:

$X_{k+1} = U_k \blacksquare X_k \blacksquare R$

with ■ treated as max-min and max-product compositions. Using the procedure of the first example, we get a model described by a set of implication statements (there are nine clusters):

If the control is U_1 *and the state is* X_1, *then the state is* X_1 *with possibility 0.59,*

If the control is U_1 *and the state is* X_2, *then the state is* X_2 *with possibility 0.44,*

If the control is U_1 *and the state is* X_3, *then the state is* X_3 *with possibility 0.39,*

If the control is U_9 *and the state is* X_1, *then the state is* X_2 *with possibility 0.78,*

If the control is U_9 and the state is X_2, then the state is X_2 with possibility 0.97,

If the control is U_9 and the state is X_9, then the state is X_9 with possibility 0.67.

The non-fuzzy result of the model give us $Q_{av} = 0.103$ (for max-min compositions) and $Q_{av} = 0.036$ (for max-product composition), shown in fig. 5.11.

5.3.10 Distributed modeling

The main trust of distributed modeling is to develop a fuzzy model that is highly distributed and operates as ensemble of logical coupled processing units. Each of them operates as an autonomous computing structure and is equipped with its own dynamics. These units can be realizing in several ways; in our discussion we will concentrate on the use of logic processor.

A high level of autonomy in this class of modeling is achieved by association each processor with an individual variable of the system. The dynamic behavior if the system results as a sequence of interactions between its variables (processor) that are carried out in either a cooperative or a competitive manner. These interactions are reflected by the excitatory or inhibitory connection set up for the logic processors. The strength of an interaction is modeled by assigning different numerical value to the corresponding connections of the processors.

The dynamics of each of the processor can be effortlessly modeled by realizing different feedback loops between the variables of the processor.

The underlying schematic structure within which the paradigm of distributing modeling is realized is portrayed in fig. 5.12. Note that in the addition to the internal links between the processor, some of them can also be exposed to various inputs from the environment.

The development of the distribute model is realized in a supervised mode. Each scenario in the collection of training events consist of the current status of the state variables (variables of the LPs) say $Q_1(k), Q_2(k), \ldots, Q_c(k)$ and inputs $I_1(k), I_2(k), \ldots, I_p(k)$ specified in the kth time instant, as well as the value of the state variable in the successive

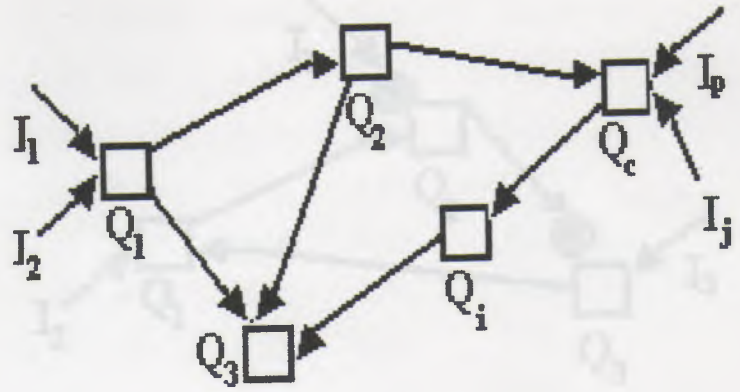


Fig. 5.12 Overall architecture of a distributed model (the nodes presents individual processes)

discrete time moment, say $Q_1(k+1), Q_2(k+1), \ldots, Q_c(k+1)$. We will assume that all $I_i s$ as well as $Q_i s$ take on numerical values situated in the unit interval. Using abbreviated vector notation

$$Q(k) = [Q_1(k) \quad Q_2(k) \cdots \quad Q_c(k)] \text{ and } \quad I = [I_1(k) \quad I_2(k) \cdots \quad I_c(k)]$$

To summarize the status of the model, the training set will then consist of triples

$$\{\mathbf{I}(k), \mathbf{Q}(k), \mathbf{Q}(k+1)\}$$

$$k = 1, 2, \ldots, N$$

Based on that, learning can be realized separately for each of the logic processors. For this purpose the algorithm describes can be fully exploited.

Within the same vain of supervised learning two essentially distinct learning situation can be distinguished, developing upon the character of the domain knowledge that is available about the system. In the first case, one has the global description of the system. This embraces all qualities dependences between the variable and the input considered in the system. The knowledge provided a prior knowledge could be easily uncorrected into the model by translating it in direct into the form of relevant connection betweens the logical processor. Example architecture is shown in fig 5.13.

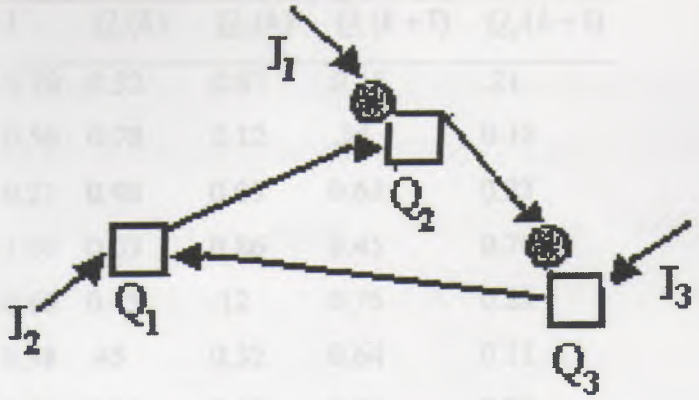


Fig 5.13 an example of distributed model with qualities links formed based on the available initial qualities domain knowledge (dots are used it summarize inhibitory connections)

This preliminary knowledge definitely eases and improves efficiency of learning, i.e. its speed, since only those connections indicated need to be qualified numerically.

On the other hand if the available knowledge about the structure is not given in detail or becomes incomplete or unreliable, one should rather concentrate on a fully connected architecture as a primary topology and carry out learning in this context. Some of the weak connection can be eliminated afterwards through building the core or Boolean approximations of the processors. As the learning is performed individually for each of logic processor, these needs no be the same with regard to the details of their internal architecture.

Example 5.4

This example illustrates how the design of the distributed model proceeds. We will start with a collection of the training data summarized in tabular.

We will assume no initial knowledge about the structural relationships between the variables so that the learning should be carried out in a fully connected topology. The performance of learning will be expressed as a sum of squared errors:

$$Q_j = \sum_{k=1} \left(t_k - LP_j(Q, I, connections)\right)^2 \quad , \text{ j} = 1, 2$$

I	$Q_1(k)$	$Q_2(k)$	$Q_1(k+1)$	$Q_2(k+1)$
0.70	0.32	0.87	0.45	.21
0.56	0.78	0.12	.34	0.18
0.21	0.98	0.05	0.65	0.23
1.00	0.03	0.86	0.45	0.78
0.60	0.45	.12	0.76	0.23
0.98	.45	0.32	0.64	0.11
0.43	0.34	0.65	0.84	0.05

The value of Q_j will be minimized separately for each of the processors. For both the processors the number of AND neurons situated in the hidden layer is 6. The results of learning are shown in Fig. 5.14.

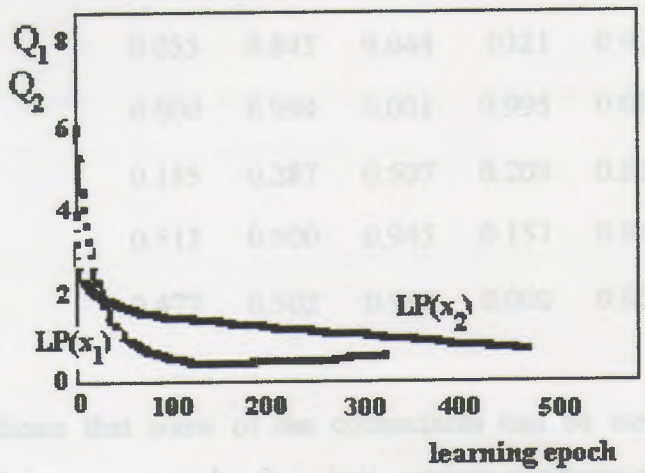


Fig. 5.14 The values of Q1, Q2 in the course of learning (α=0.1, t-norm: product, s-norm: probabilistic sum)

The results of learning expressed in terms of the connections (weights) of the processors are:

Lp1

Output-hidden layer:	0.075	0.081	1.000	0.089	0.850	1.000
Input-hidden layer:	0.328	0.167	0.368	0.426	0.057	0.467
	0.113	0.848	0.012	0.265	0.991	0.327
	1.000	1.000	1.000	0.939	0.010	0.260
	0.225	0.383	0.388	0.210	0.772	0.268
	1.000	0.598	0.969	0.008	0.933	1.000
	1.000	1.000	1.000	0.419	0.007	0.450

Lp2

Output-hidden layer:	0.000	0.000	0.085	0.000	.0000	0.065
Input-hidden layer:	0.318	0.160	0.371	0.424	0.081	0.474
	0.055	0.845	0.044	.0321	0.909	0.378
	0.000	0.994	0.001	0.995	0.000	0.995
	0.185	0.287	0.507	0.209	0.839	0.207
	0.513	0.000	0.945	0.157	0.840	0.856
	0.477	0.502	0.999	0.000	0.809	0.000

They clearly indicate that some of the connections can be easily eliminated. The approximation of the processors by their core versions produces small values of error for the optimal threshold levels; the plots of the surfaces of the approximation error displayed with respect to the threshold levels μ and λ support this finding.

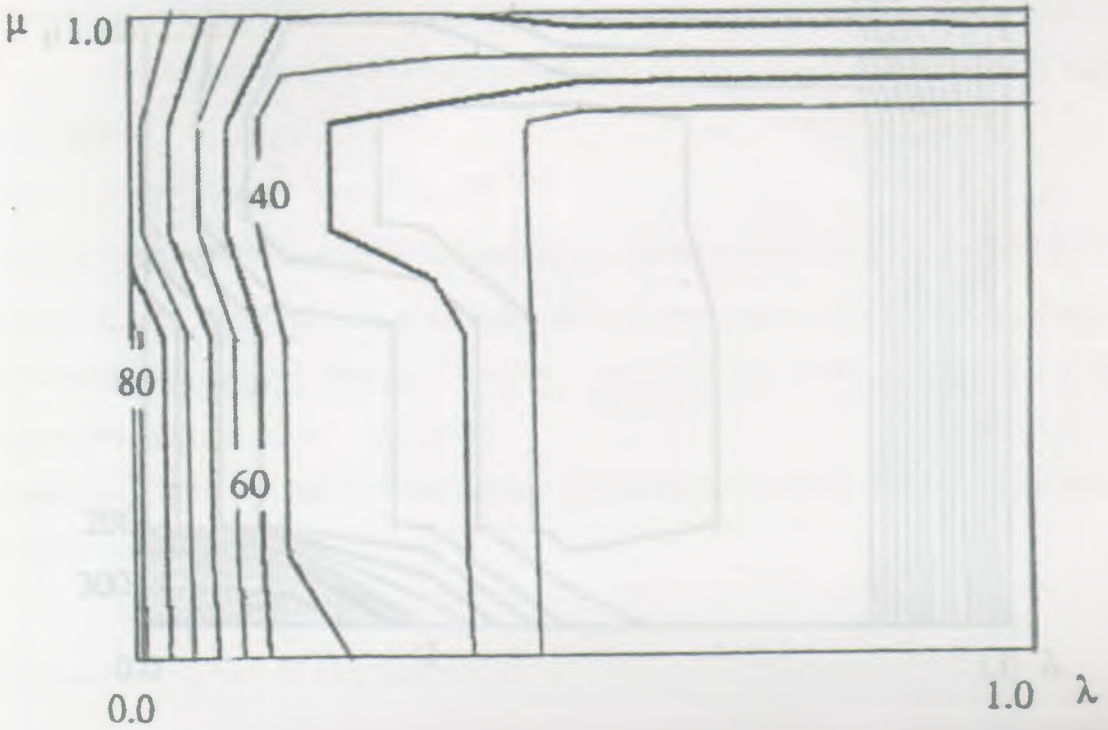


Fig. 5.15 approximation error produced by the core structure of the logic processor for the inputs uniformly distributed over [0,1]

The optimal Boolean approximation gives rise to the higher values of the approximation error, see fig 5.16.

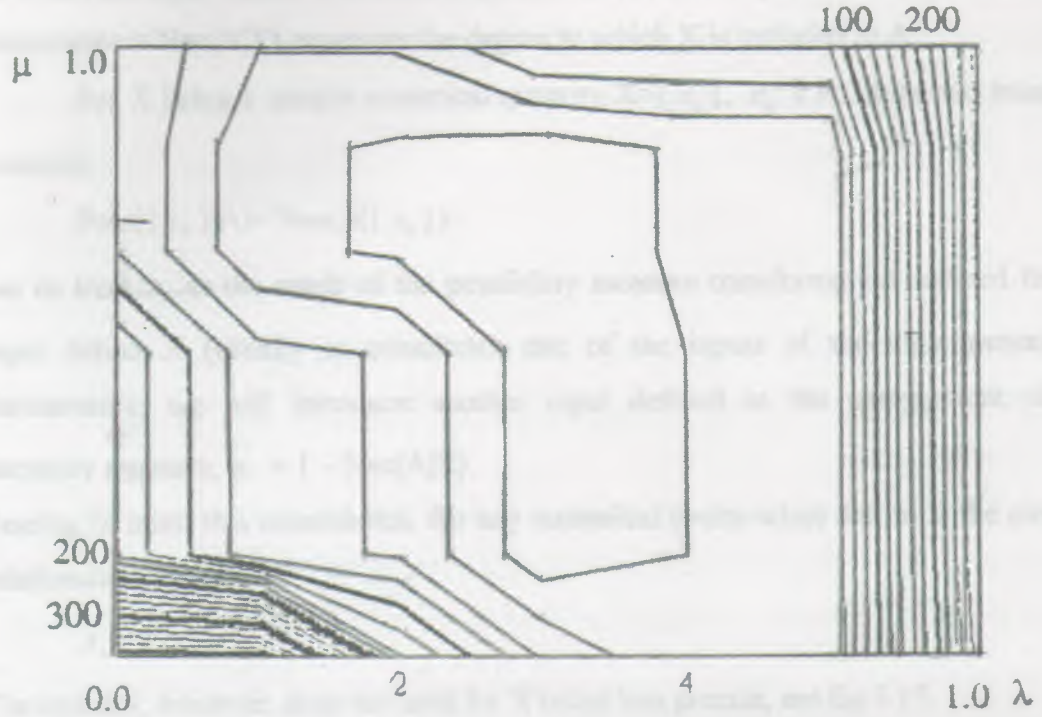


Fig. 5.16 Approximation error produced by the Boolean structure of the logic processor for the input uniformly distributed over [0,1]

The resulting formulas read as

$$Q_1(k+1) = (\overline{Q_1}\, AND \overline{Q_2}) OR (\overline{I}\, AND \overline{Q_1}\, AND \overline{Q_2}) = \overline{Q_1}\, AND \overline{Q_2}$$

$$Q_2(k+1) = I \;\; AND \;\; Q_2 \;\; AND \;\; \overline{Q_1}$$

As stated, the primary aim of distributed modeling is to capture relationships between the system variables formulated at the logical level. This implies that all of the processors when combined together describe interactions, and subsequently the dynamics, of the system as it has been manifested at the level of these labels. Obviously, the linguistic labels with different levels of granularity could give rise to quite diversified dynamical patterns of the model.

Depending on applications, different interfaces between the model and the environment are worth considering. In particular, the one with the possibility-necessity transformation is interesting as being capable of expressing imprecision about the input numerical information. That the possibility measure Poss(X|A) characterizes an extent

to which the input datum X and the linguistic label A overlap (coincide). The necessity measurement Nec(A|X) expresses the degree to which X is included in A.

For X being a simple numerical quantity X={ x_0 }, $x_0 \in$ R, these two measures coincide,

Poss({ x_0 }|A)= Nec(A|{ x_0 })

Let us treat x_i as the result of the possibility measure transformation realized for the input datum X (usually xi constitutes one of the inputs of the logic processor). Furthermore, we will introduce another input defined as the complement of the necessity measure, $\bar{x}_i = 1 - \text{Nec}(A|X)$.

Bearing in mind this coincidence, for any numerical (point-wise) datum X the obvious relationship is preserved:

$$x_i + \bar{x}_i = 1$$

The equality, however, does not hold for X being less precise, see fig 5.17.

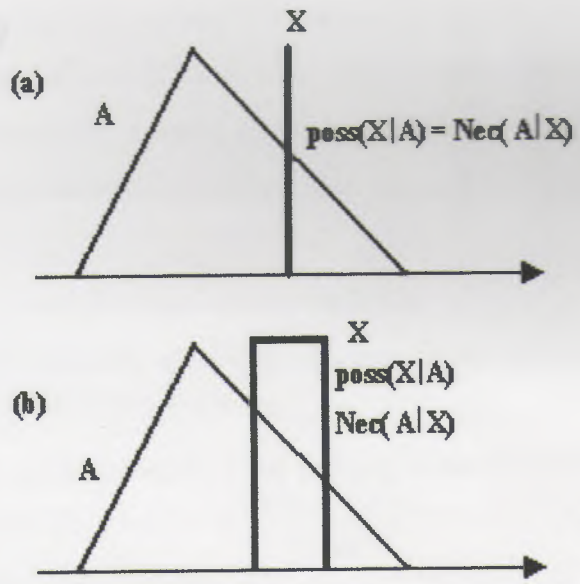


Fig. 5.17 Uncertainty in a numerical datum X conveyed by the possibility and necessity measures; (a) point-wise datum X, (b) interval-valued datum X.

For the internal-valued information the above equality is violated, yielding $x_i + \bar{x}_i > 1$. This inequality results from a straightforward observation:

Poss(X|A) + (1-Nec(A|X) = 1 + Poss(X|A) – Nec(A|X) ≥ 1

Since Poss(X|A) ≥ Nec(A|X). The higher the uncertainty level (broader interval X), the more significant the departure from the original constraint. In the limit case, uncertain information produces $x_i + \bar{x}_i = 2$. This holds, for instance, when a normal fuzzy set A is contained in the interval-valued X, $A \subset X$. the value $\xi \in [0,1]$ resulting from the expression

$$x_i + \bar{x}_i = 1 + \xi$$

can be viewed as an indicator of imprecision (uncertainty) of X with respect to A. it should be stressed that ξ is a relative measure of uncertainty expressed in the context of A.

Bearing in mind that this factor of uncertainty could also occur in the training data set we can describe a way in which it is transformed by the logic processor by augmenting its basic one-output structure by an additional output node characterizing $\bar{y}$. Then the uncertainty level produced by the processor can be expressed numerically through the values of $\eta \in [0,1]$, where

$$y_i + \bar{y}_i = 1 + \eta$$

CONCLUSION

We have provided an introduction to several techniques on how to construct fuzzy systems. We used a simple example to illustrate how several of the methods operate. The least squares method can be used to train linear systems and should be considered as a conventional alternative to the other methods (since it can sometimes be easier to implement). Gradient methods are especially useful for training parameters that enter in a nonlinear fashion. The clustering and optimal output predefuzzification method combined the conventional least squares method with the c-means clustering technique that provided for the specification of the input membership functions and served to interpolate between the linear models that were specified via least squares. Clustering based in nearest neighborhood methods helps to provide insight into fuzzy system construction.

Upon completing these three chapters, we should understand the following:

- The function approximation problem.
- How construction of models, estimators, predictors, and controllers can be viewed as a special case of the function approximation problem.
- The issues involved in choosing a training data set.
- How to incorporate linguistic information into a fuzzy system that you train with data.
- The batch and recursive least squares methods.
- How to train standard or Takagi-Sugeno fuzzy systems with least squares methods.
- The gradient algorithm method for training a standard or Takagi-Sugeno fuzzy system.
- The clustering with optimal output predefuzzification method constructing Takagi-Sugeno fuzzy system.
- The nearest neighborhood clustering method for training standard fuzzy systems.

And we have studied in depth the problem of system identification in the presence of fuzziness. After a look at where fuzzy models are useful, we have concentrated on specific areas of fuzzy modeling. Techniques applying not only fuzzy sets but also probabilistic forms of information have been discussed. Identification is viewed as an

iterative, three-phase procedure comprising a sequence of stage: (i) determination of the structure of the model, (ii) estimation of its parameters, and (iii) validation of the constructed model. The scheme is of a general nature: however, fuzzy models need several novel techniques if they are to work within its framework. This is especially important in parameter estimation, where careful attention must be paid to the structure of the data set (input-output fuzzy sets). The notion of identifiability of the data set enables us to establish a rationale for finding the most readily estimated one. We have examined the evaluation of the fuzzy model by the fuzzy measure and the fuzzy integral and by confidence intervals. This strong tool for evaluation allows the user to decide whether a consecutive iteration loop is required. The methods proposed for model validation can be used to elicit its behavior in a different application, such as control or predication. We have studied the basic principles of identification before moving up to the relevant algorithms, thereby filling a gap in the literature.

REFERENCES

[1] Box, G.E., & Jenkins, G.M. 1970. 'Time Series Analysis: Forecasting and Control'. Holden Day, San Francisco, USA

[2] Duda, R.O., & Hart, P.E. 1974. 'Pattern Recognition and Scene Analysis'. John Wiley, London, England

[3] Eyckhoff, P. 1974. 'System Identification: Parameter and State Estimation'. John Wiley, London, England

[4] Gottwald, S., & Pedrycz, W. 1986. 'On the suitability of fuzzy models: an evaluation through fuzy integrals'. *Int. J. Man-Match. Stud.*, 24, pp. 141-151

[5] Hirota, K., & Pedrycz, W. 1980. 'On identification of fuzzy system under the existence of vagueness'. *In* 'Summery of Papers on General Fuzzy Problems'. 6, pp. 37-40

[6] Hirota, K., & Pedrycz, W. 1982. 'Fuzzy system identification via probabilistic sets'. *Inf*. Sci. (USA), 28, pp. 21-43

[7] Hirota, K., & Pedrycz, W. 1983. 'Analysis and synthesis of fuzzy systems by the use of probabilistic sets'. *Fuzzy Sets & Syst.*, 10, pp. 1-13

[8] Pedrycz, W. 1981. 'An approach to the analysis of fuzzy systems'. *Int. J. Contr.*, 3, pp. 403-421

[9] Pedrycz, W. 1984. 'Construction of fuzzy relational models'. *In* 'Prpc. Cybernetics & Systems Res'. (Ed. R. Trappl). North Holland, Amsterdam, pp. 545-549

[10] Pedrycz, W. 1984. 'On identification algorithm in fuzzy relational systems'. Fuzzy Sets & Syst., 13, pp. 153-167

[11] Pedrycz, W. 1985. 'Design of fuzzy control algorithms with the aid of fuzzy models'. *In* 'Industrial Application of Fuzzy Control' (Ed. M. Sugeno). North Holland, Amsterdam, pp. 153-173

[12] Pedrycz, W. 1986. 'Structured fuzzy models'. *Cybern. & Syst.*, 16, pp. 103-117

[13] Tong, R.M. 1977. 'A control engineering review of fuzzy systems'. *Automatica*, 13, pp. 559-569

[14] Tong, R.M. 1980. 'The evaluation of fuzzy models derived from experimental dta'. *Fuzzy Sets & Syst.*, 4, pp. 1-12

[15] Tong, R.M., Beck, M.B., & Latten, A. 1980. 'Fuzzy control of the activated sludge wastewater treatment process'. *Automatica*, 16, pp. 595-701

[16] Zadeh, L.A. 1971. 'Toward a theory of fuzzy systems'. *In* 'Aspects on Network and System Theory' (Eds. R.E. Kalman & N. De Claris). Holt, Rinehart, Winston, New York, USA, pp. 209-245

[17] Zadeh, L.A. 1973. 'Outline of a new approach to the analysis of complex systems, and decision processes'. *IEEE Trans. Syst., Man & Cybern.,* 1, pp. 28-44

[18] Tulleken, 1993; Johansen, 1996.

[19] Schubert, 1994; Thompson and Kramer, 1994; Psichogios and Ungar, 1992; van Can, et al., 1997.

[20] Gancho Vachkov and Toshio Fukuda. "*Simplified Fuzzy Model Based Identification of Dynamical Systems*". *International Journal of Fuzzy Systems, Vol. 2, No. 4, December. 2000.*

[21] M. Hadjili, A. Lendasse, V. Wertz and S. Yurkovish. "*Identification of Fuzzy Models for A Glass Furnace Process*". Proceeding of the 1998 IEEE, International Conference on Control Applications.

[22] P. Srinivasa Babu, Armidan Ghosh, Sachchidanand. "*Fuzzy Identification and Control of A Class of Nonlinear Systems*". Indian Institute of Technology, Kanpur, INDIA.

[23] A. Johansen and Robert Babuska. "*On Multi-Objective Identification of Takagi-Sugeno Fuzzy Model Parameters*". Norwegian University of Science and Technology, 7491 Trondheim, Noorway.

[24] P. Krause, A. Krone, T. Slawinski. "*Fuzzy System Identification by Generating* and *Evolutionary Optimizing Fuzzy Rule Bases Consisting of Relevant Fuzzy Rules*". University of Dortmund, D-44224 Dortmund.