Rule-Base Structure Identification in an Adaptive-Network-Based Fuzzy Inference System

Chuen-Tsai Sun

***Abstract*—Fuzzy rule-base modeling is the task of identifying the structure and the parameters of a fuzzy IF–THEN rule base so that a desired input/output mapping is achieved. Recently, using adaptive networks to fine-tune membership functions in a fuzzy rule base has received more and more attention. In this paper we summarize Jang's architecture of employing an adaptive network and the Kalman filtering algorithm to identify the system parameters. Given a surface structure, the adaptively adjusted inference system performs well on a number of interpolation problems. We generalize Jang's basic model so that it can be used to solve classification problems by employing parameterized t-norms. We also enhance the model to include weights of importance so that feature selection becomes a component of the modeling scheme. Next, we discuss two ways of identifying system structures based on Jang's architecture. For the top-down approach, we summarize several ways of partitioning the feature space and propose a method of using clustering objective functions to evaluate possible partitions. We analyze the overall learning and operation complexity. In particular, we pinpoint the dilemma between two desired properties: modeling accuracy and pattern matching efficiency. Based on the analysis, we suggest a bottom-up approach of using rule organization to meet the conflicting requirements. We introduce a data structure, called a fuzzy binary boxtree, to organize rules so that the rule base can be matched against input signals with logarithmic efficiency. To preserve the advantage of parallel processing assumed in fuzzy rule-based inference systems, we give a parallel algorithm for pattern matching with a linear speedup. Moreover, as we consider the communication and storage cost of an interpolation model, it is important to extract the essential components of the modeled system and use the rest as a backup. We propose a rule combination mechanism to build a simplified version of the original rule base according to a given focus set. This scheme can be used in various situations of pattern representation or data compression, such as in image coding or in hierarchical pattern recognition.**

I. Introduction

THE concept of system modeling is closely related to interpolative input–output mapping, pattern classification, case-based reasoning, and learning from examples. It plays an important role in rule-based control, data compression, pattern recognition, expert systems, and multiple-objective decision processes. Conventional approaches of system modeling perform poorly in dealing with complex and uncertain systems, because it is very difficult to find a global functional or analytical structure for a nonlinear system.

Manuscript received May 15, 1991; revised August 30, 1991. This work was supported by NASA under Grant NCC-2-275, by LLNL under Grant ISCR 89-12, by the MICRO State Program under Award 90-191, by MICRO Industry, and by Rockwell under Grant B02302532.

The author is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050 R.O.C.

IEEE Log Number 9201461.

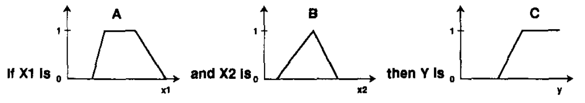


Fig. 1. A typical fuzzy rule.

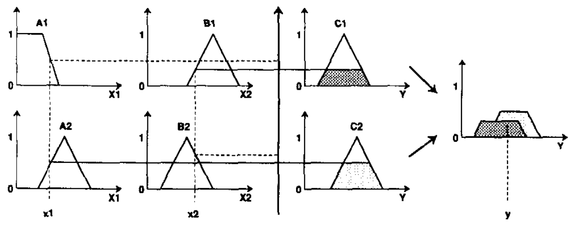


Fig. 2. Approximate reasoning.

On the contrary, a fuzzy inference system uses fuzzy IF–THEN rules to describe a system. Fig. 1 gives a typical example of a fuzzy rule, where X_1 and X_2, are input variables, Y is an output variable, and A, B, and C are linguistic terms [31] characterized by appropriate membership functions [30].

Consequently, a fuzzy rule gives a meaningful expression of the qualitative aspects of human reasoning. Based on pattern matching between rule antecedents and input signals, a number of fuzzy rules are triggered in parallel with various values of firing strength. Individually invoked actions are combined together with a defuzzification mechanism to give a single output. The inference process is called approximate reasoning, as illustrated in Fig. 2.

While there are many ways of modeling a fuzzy inference system, such as the ones using fuzzy relation equations [14], [18] and those represented as fuzzy associative memory (FAM) banks [8], [9], [27], this paper follows the approach proposed by Sugeno and his colleagues [24], [26], which is based on the idea of finding a set of local input–output relations (rules) to describe a system. Fuzzy rule-based systems have found many successful applications. Further, each fuzzy rule can be considered as a localized receptive field [12], which is biologically applaudable in human perception.

We can perform fuzzy modeling by extracting knowledge from human experts and by transforming the expertise into rules and membership functions. The resulting system can then be tuned by monitoring its performance through trial and error.

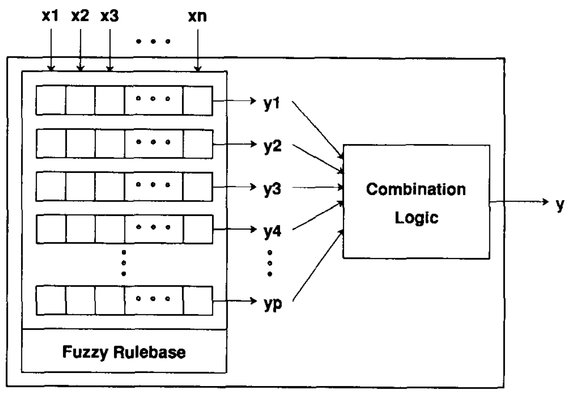


Fig. 3. A fuzzy inference system.

However, depending on human introspection and experience results in some severe problems. First of all, even when human experts exist, their knowledge is often incomplete and episodic rather than systematic. Moreover, there is no formal and effective way of knowledge acquisition. For example, determining the proper number of rules and partitioning the input feature space for the most part remain arts. Finally, we want the system to have learning ability to update and fine-tune itself based on newly arriving information.

As a result, researchers have been trying to automate the modeling process based on numerical training data. The goal is to establish a rule-based system of the type shown in Fig. 3. Each rule, representing a local description of the system's behavior, is composed of an antecedent part (condition) and a consequent part (action). The antecedent part is a fuzzy pattern of multidimensional input features; the consequent part is a linear combination of inputs. The modeling task can be divided into two parts: structure identification and parameter identification. The former is related to finding a suitable number of rules and a proper partition of the feature space; the latter deals with the adjustment of system parameters, such as the membership functions, linear coefficients, and combination logic.

Recently, Jang [5] proposed a method of using adaptive networks and Kalman filters for parameter identification and obtained impressive simulation results. This architecture is quite general in that it can be used in both interpolation and classification problems. It can also be modified to incorporate advanced fuzzy pattern matching techniques such as weights of importance and fuzzy quantifiers in an approximate reasoning process. We show simulation data to verify the effectiveness of these generalizations. Our further discussion on structure identification is based on this model.

Jang's approach assumes a predetermined structure, i.e., a gridlike partition of the feature space. We show, in terms of topological and computational complexity, that the assumption is no longer true when the system to be modeled is sophisticated and the number of input variables becomes large. Several multidimensional data structures for partitioning the feature space are summarized in this paper. We propose a hill-climbing method based on *k–d* trees to solve the structure identification problem. Two objective functions, a density measure and a typicality measure, are used in the partition process to find a proper starting point for the parameter identification phase. The entire modeling scheme is applied to a temperature prediction problem and gives satisfying results.

However, when we face a really complicated system, a simple, global, and effective partition is difficult to find. Now we encounter the dilemma between modeling accuracy and learning/operation efficiency. We propose a method of rule organization to solve this problem. It employs the concept of divide and conquer to achieve modeling accuracy with many small rules, each covering a local region. Then we build a binary fuzzy boxtree out of the rules based on a similarity measure between their antecedent patterns. We claim that a branch-and-bound algorithm can do the pattern matching job for firing appropriate rules with logarithmic efficiency so that the large number of rules will cause no trouble for the entire scheme. Moreover, to maintain the advantage of fuzzy rule-based systems in parallel processing, a parallel algorithm is proposed to meet the requirement.

Another alternative to cope with sophisticated systems is to use rule combination so that we can apply a simplified rule-base. When there is a focus area in the application domain, this is a promising approach. We give an algorithm for rule combination with respect to a boxtree and show how to invoke rule bases with various granularities in an interpolation problem.

II. Learning with Adaptive Networks

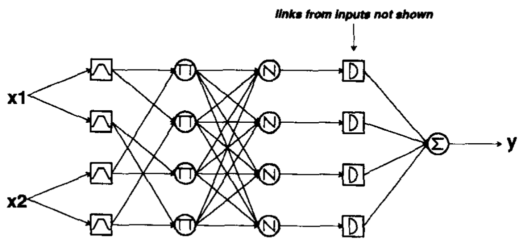
Now we consider the problem of parameter identification of a fuzzy rule-based inference system. As pointed out by Zadeh [33], when the rule-base structure can be formulated, we can use adaptive networks to calibrate the membership functions. Various approaches have been proposed in this direction, as in [25] and [29]. Our discussion and the development of structure identification mechanisms in the following sections are based on Jang's method [5] because of its comprehensibility and effectiveness.

An adaptive network is a multilayer feedforward network in which each node performs a particular function (node function) based on incoming signals and a set of parameters pertaining to this node. When the set of parameters is empty, we use a circle to denote the node; otherwise we use a square. The type of node function may vary from node to node with the choice of node function depending on the overall function that the network is designed to carry out.

Fig. 4 demonstrates Jang's architecture with two input variables, x_1 and x_2, and one output variable, y. Each input is represented as two linguistic terms; thus we have four rules. The nodes in the same layer have the same type of node function.

Each node at layer 1 (membership) is associated with a parameterized bell-shaped membership function represented as

$$\mu_A(x_i) = \frac{1}{1 + \left[\left(\frac{x_i - c_i}{a_i}\right)^2\right]^{b_i}}, \tag{1}$$



Layer	1	2	3	4	5
Node	L*V	R	R	R	1
Parameter	3	0	0	V+1	0

Fig. 4. An adaptive-network-based fuzzy inference system. The table at bottom shows the number of nodes and the number of parameters of each node at each layer. V is the number of input variables; L is the number of linguistic terms for each input; and R is the number of rules. In this case, $R = L_V$.

where x_i is one of the input variables, A is the linguistic term associated with this node function, and $\{a_i, b_i, c_i\}$ is the parameter set. The parameters are tuned with back-propagation, a gradient descent method, in the learning process based on a given training data set. Each node at layer 2 (conjunction) multiplies the incoming signals and sends the product out. The output signal corresponds to the firing strength of a rule.

The ith node at layer 3 (normalization) calculates the ratio of the ith rule's firing strength to the sum of the firing strengths of all the rules, i.e., the relative portion of the ith rule in the final result. A node at layer 4 (defuzzification) calculates a linear combination of input signals, i.e.,

$$d_0 + d_1 * x_1 + d_2 * x_2, \tag{2}$$

and multiples the result with the weight coming from layer 3. During the learning process, the d_i's are adjusted using a Kalman filter [7] to minimize the mean square error between the calculated output and the desired output. Finally, the single node at layer 5 (summation) produces the weighted sum of the output signals coming from invoked rules.

Note that in Jang's method, the firing strength of each fuzzy rule is calculated as the product of the membership values in the premise part, the consequence of each rule is a linear combination of inputs, and the final output is obtained as the weighted average of each rule's consequence. However, other types of fuzzy reasoning method [10] can also be implemented with this model. The number of nodes at each layer and the number of parameters associated with each node are summarized in a table shown at the bottom of Fig. 4.

The effectiveness of modeling using Jang's model is shown in [5]. It gives not only a meaningful representation of a system, but also a comprehensive network topology for the learning process. Further, Jang's model is a special configuration of layered adaptive networks with localized receptive fields, which are supported by biological evidence [12] and are relatively efficient to learn [15]. Another advantage of this model is its simplicity in terms of connections between two layers, which is a critical factor in VLSI implementation [3], [11]. Moreover, the basic model can be generalized in several directions to solve various problems, as discussed in the next section. Consequently, we choose Jang's model as the basis for structure identification as well as rule organization. Our analysis and design are largely based on architectural and computational complexity.

III. Generalizations of Jang's Basic Model

From Fig. 4, it is easy to see that the topological complexity of Jang's model is $O(R)$ or $O(L^V)$. This results from the assumption that the feature space if uniformly partitioned along each input dimension. Since the size of the network grows exponentially as the number of inputs increases, obviously the learning and operation efficiency will be poor when the architecture is used to model a complicated system with many variables. Thus, the first generalization of Jang's model is to abandon the uniform-partition assumption and consider an arbitrary rule-base configuration. When the configuration, i.e., a proper partition of the feature space, is not known at the beginning of the modeling process, we have to do structure identification before the parameter calibration can be manipulated. Structure identification is discussed in the next section.

Moreover, the discussion up to this point is based on the implication that the input variables are of equal importance. However, in applications of automatic control, pattern classification, or multicriteria decision making, this assumption is usually not true. In other words, a general modeling scheme should include feature selection. We can enhance the basic model by introducing the concept of weight of importance in fuzzy pattern matching to fulfill this requirement.

Assume each input variable x_i is associated with a weight of importance $w_i \in [0, 1]$. We generalize the result of (1) to a weighted degree of match, denoted by s_i:

$$s_i = 1 - w_i * (1 - \mu_A(x_i)). \tag{3}$$

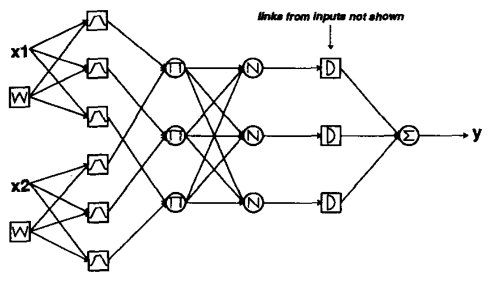
Note that the less important a variable is (w_i small), the smaller the role that s_i plays in the multiplication at the next layer in the adaptive network. On the other hand, if a variable is of full importance ($w_i = 1$), s_i reduces to $\mu_A(x_i)$.

Fig. 5 gives the configuration of the generalized model with two inputs and three rules. Each weight node at layer 1 has the weight of importance as its only parameter. The initial value of weight is defaulted at 1.0 but can be assigned to any value in [0, 1] by users based on their heuristic judgment. Note that the architectural complexity is now of $O(R * V)$ with R independent of V.

During the learning process, once a weight of importance is stabilized below a certain threshold value, the corresponding variable is considered unimportant in the system to be modeled. Thus, the variable can be neglected and a simplified structure/parameter identification process can be resumed to find an even better solution.

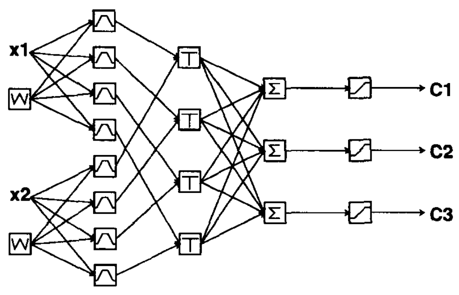
IV. Modeling Scheme for Classification Problems

Jang's model assumes that an output value can be represented as a particular linear combination of inputs within each region in the feature space. The overlapping among regions provides the natural smoothness for the input-output



Layer	1	2	3	4	5	6
Node	V	R*V	R	R	R	1
Parameter	1	3	0	0	V+1	0

Fig. 5. Generalized configuration with weights of importance.



Layer	1	2	3	4	5
Node	V	R*V	R	C	C
Parameter	1	3	1	R	1

Fig. 6. An adaptive-network-based fuzzy classifier. C is the number of fuzzy classes.

mapping. This characteristic of Jang's model makes it suitable for applications such as automatic control and time series prediction.

Another important category of geometric learning, classification, has some different properties. Generally speaking, a classifier partitions the feature space based on labeled training data. In the context of fuzzy classification see [16], [17], [32] classes are overlapping and each training data item is associated with numbers in [0, 1] representing degrees of belonging, one value for each class. A modified network based on Jang's scheme can be used as a fuzzy classifier (see Fig. 6).

There are two input variables and four rules in this example and we assume that the training data are categorized by three fuzzy classes. We still employ weights of importance for feature selection at Layer 1. Layer 2 calculates the membership values for rule premises as before.

In most pattern classification and query-retrieval systems, the conjunction operator plays an important role and its interpretation changes across contexts. Since there is no single operator that is suitable for all applications, we use parameterized t-norms at layer 3 to cope with this dynamic property of classifier design. Bonissone provided a detailed discussion on t-norms and their parameterized versions (see [2]). For example, we can use Hamacher's t-norm:

$$T_H(x_1, x_2, \gamma) = \frac{x_1 x_2}{\gamma + (1-\gamma)(x_1 + x_2 - x_1 x_2)}, \tag{4}$$

where x_i's are the operands and γ is a nonnegative parameter.

In some other applications, e.g., see [35], features are combined in a compensatory way. For these situations, mean operators [28] are more appropriate than conjunctive operators. To find a good mean operator for a certain system, we can also implement a parameterized operator and use training data to calibrate it. For instance, we can use the one proposed by Dyckhoff and Pedrycz:

$$M_{DP}(x_1, x_2, \gamma) = \frac{(x_1^\gamma + x_2^\gamma)^{1/\gamma}}{2}, \tag{5}$$

where $\gamma \geq 1$. Note that we can use a parameterized operator for each node in layer 3 or employ a single one for the whole layer. Whether an operator is local or global depends on applications. Moreover, a parameterized fuzzy quantifier [6] can also be introduced into this picture based on the same concept.

We take the linear combination of the firing strengths of the rules at layer 4 and apply a sigmoidal function at layer 5 to calculate the degree of belonging to a certain class. Again, the parameters in layers 4 and 5 can be adjusted by a Kalman filter and the membership functions at layer 2 are fine-tuned by back-propagation.

We applied this learning scheme on an Iris classification problem to find the mapping between four input variables (sepal length, sepal width, petal length, and petal width) and three classes. In order to evaluate the performance, we define an average error (APE) as

$$APE = \frac{\sum_{i=1}^{P} |T(i) - O(i)|}{\sum_{i=1}^{P} |T(i)|} * 100\% \tag{6}$$

where P is the number of training data and $T(i)$ and $O(i)$ are the ith desired output and the calculated output, respectively. After 200 epochs of learning we obtained a modeling error $APE = 1.984\%$, a fairly good result.

V. Feature Space Partitioning

We justified the need for structure identification in the previous discussion. Now we describe various methods of achieving this goal and their limitations. These methods can be divided into two categories: feature space partitioning and rule organization. We discuss feature space partitioning here and will cover rule organization in the next section.

As mentioned before, a rule-base structure can be viewed as a partition in the multidimensional feature space. Fig. 7 shows several data structures frequently used in modeling a multidimensional sophisticated system. These structures possess a common property: they are composed of fuzzy hyper-rectangles, or fuzzy boxes. The antecedent part of a fuzzy rule can be considered as a fuzzy hyper-rectangle in the

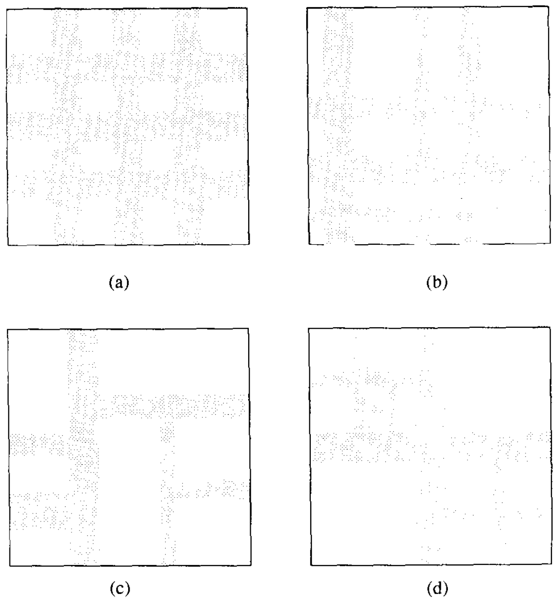


Fig. 7. Four fuzzy multidimensional data structures: (a) fuzzy grid, (b) adaptive fuzzy grid, (c) fuzzy k–d tree, (d) multilevel fuzzy grid. In two-dimensional space, the structure in (d) is also called a fuzzy quad tree.

multidimensional feature space. Consequently, the structures in Fig. 7 are all candidates for modeling a fuzzy rule base.

Wang and Mendel [27] used fuzzy grids [Fig. 7(a)] as their data structure for FAM banks. They proposed a deterministic procedure to decide the mapping between a fuzzy region and an output linguistic term. Since the procedure is deterministic, there is no learning involved in their method. The performance depends entirely on the definition of the fuzzy grid. In general, the finer the grid, the better the modeling performance becomes. However, there are two major drawbacks when we partition a certain feature dimension into a large number of regions. First, the operation efficiency suffers. Second, it becomes likely that some regions will not cover any training data and will remain undefined. Karr [8], [9] used the same rule-base structure but he employed genetic algorithms to optimize the mapping.

Jang's basic model is built on adaptive fuzzy grids [Fig. 7(b)]. At the beginning of learning, a uniformly partitioned grid is taken as the initial state. As the parameters in the premise membership functions are adjusted, the grid evolves. The gradient descent method optimizes the location and size of the fuzzy regions and the degree of overlapping among them. Two problems exist in this scheme. First, the number of linguistic terms for each input variable is predetermined and is highly heuristic. Second, the learning complexity suffers an exponential explosion as the number of inputs increases.

Fuzzy grids give the most restricted structures; on the other extreme, fuzzy clustering algorithms [1], [22] based on training data result in the most flexible partitions. However, this approach has its own problems. First of all, the resulting structures are not necessarily hyper-rectangles; they need refinement. To map a fuzzy cluster to a set of bell-shaped functions, we can consider c_i in (1) as the coordinate of the cluster center, $\vec{h}$, in the ith dimension; meanwhile, a_i is assigned to the value of cluster radius, i.e., the longest distance from the center to a point, $\vec{p_i}$, with nonzero membership in the ith dimension. The value of b_i is interpreted as a slope and can be determined as a linear function of the membership of the boundary point $\vec{p_i}$.

The second weak point of clustering algorithms is that the cost tends to be high, because the efficiency of convergence is not guaranteed. Third, the total number of rules (clusters) is predetermined in these algorithms. When the resulting partition is not good enough and we want to increase the number of rules, we have to rerun the clustering algorithm from the beginning.

The essential point here is that in the context of adaptive network training, we do not need to find a perfect clustering since our goal is just to find a satisfiable initial state for the adaptive network to tune. In other words, as we have verified the validity of the parameter identification mechanism using adaptive networks, it makes no sense to spend a lot of time in optimizing the cluster criteria, no matter what the objective functions might be.

Thus, we adopted an intermediately flexible partitioning, the *fuzzy k–d trees* [Fig. 7(c)], for structure identification. In the following, we explore several ways of providing a feature space partition based on fuzzy k–d trees.

A k–d tree results from a series of *guillotine cuts*. By a guillotine cut, we mean a cut which is made entirely across the subspace to be partitioned; each of the regions so produced can then be subjected to independent guillotine cutting. At the beginning of the ith iteration step, the feature space is partitioned into i regions. Now another guillotine cut is applied to one of the regions to further partition the entire space into $i + 1$ regions.

There are various strategies to decide which dimension to cut and where to cut it at each step. Some are based merely on the distribution of training examples; others take the parameter identification methods into consideration. We list and briefly discuss several strategies in the following before introducing a hill-climbing method based on fuzzy clustering objective functions.

1) Balanced-Sampling Criterion: The simplest tactic is to cut the dimension in which the training data associated with the region are most spread out and to cut it at the median value of those samples in that dimension [4]. The expected shape of the regions under this procedure is asymptotically cubical because the long dimension is always cut. In general, this method produces homogeneously distributed localized receptive fields.

2) Information Gain: The cutting procedure can be viewed as a method of building a decision tree. Quinlan [19] proposed a method based on information theory and defined the concept of information gain at each branch. Therefore, the rule of thumb is to choose a cut with the most information gain at each step.

3) Regional Linearity: This strategy is suitable when the consequence part of a rule is represented as a linear combination of the input features, as in Jang's basic model. Conceptually, we want to use a hyperplane to approximate the training samples in a region and minimize the mean square error. Thus, we can apply a Kalman filter to identify a set of linear coefficients and the cut resulting in the least error will be selected under this criterion.

4) Direct Evaluation: The most direct, but inefficient, way to evaluate a partition is to feed the resulting structure into the parameter identification phase and use the final performance to choose the best cut. Sugeno and Kang [24] used this approach together with a large number of heuristics to find the proper structure.

The above methods are all crisp partitions so the result need to be fuzzified. Furthermore, the evaluation functions used in these hill-climbing algorithms are either too shallow, without considering the need of parameter identification, or too deep, thus bothered by a credit assignment problem. We propose as a compromise using two fuzzy clustering objective functions, a typicality measure and a density measure. The basic assumption of this approach is simply that a good fuzzy rule is usually represented by a cluster which has a prototypical center and a strong support from the samples.

VI. Hill-Climbing with Fuzzy Clustering Objective Functions

Our method is still an n-step hill-climbing approach, where n is the desired number of rules, because of considerations of efficiency. At each step a fuzzy set is defined for each cluster i with the following membership function, which will be used in the objective functions:

$$\mu_{ik} = \prod_{j=1}^{V} \frac{1}{1 + \left[\left(\frac{x_{kj} - c_{ij}}{a_{ij}}\right)^2\right]^{b_{ij}}}, \tag{7}$$

where μ_{ik} denotes the membership of the kth point $(\vec{x_k})$ in the ith cluster, V is the number of variables, x_{kj} is the jth coordinate of $\vec{x_k}$, and Π stands for fuzzy conjunctive operator. Calculation of the parameters $a_{ij}, b_{ij}, \text{and}\, c_{ij}$ is dependent on the hyper-rectangle defined by a cluster resulting from guillotine cuts. Let $\vec{h_i}$ be the physical center of the ith hyper-rectangle; then c_{ij} is defined as the jth coordinate of $\vec{h_i}, a_{ij}$ is calculated as half the length along the hyper-rectangle's jth dimension and b_{ij} is determined by the desired degree of overlapping between fuzzy regions. At the end of the structure identification phase, the a's, b's, and c's are fed into the adaptive network as the initial parameters.

Now we define two objective functions for the best-first search process. As analyzed by Bezdek [1], various objective functions can suggest radically different substructures in the same data set. To achieve a meaningful structure for a fuzzy rule base, we have to select appropriate measures. In our approach, we use two objective functions: one (J_D) is a density measure; the other (J_T) is a typicality measure.

J_D was proposed by Ruspini [21]:

$$J_D = \sum_{j=1}^{P}\sum_{k=1}^{P}\left\{\left[\sum_{i=1}^{C}(\mu_{ij} - \mu_{ik})^2\right] - d_{jk}^2\right\}^2, \tag{8}$$

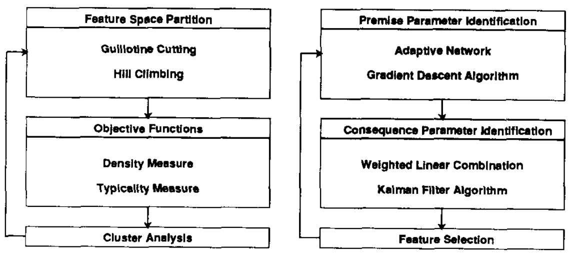


Fig. 8. A general fuzzy modeling scheme: (a) structure identification; (b) parameter identification.

where P is the number of training data, C is the number of clusters (rules), d_{jk} is the distance (or the measure of dissimilarity) between sampling points j and k, and μ_{ij} is the membership of point j in cluster i. As pointed out by Ruspini, J_D is a measure of cluster quality based on local density, because J_D will be small when terms in (8) are individually small; in turn, this will occur when close pairs of points have nearly equal fuzzy memberships in the C clusters.

J_T is a variation of the least square functional proposed by Bezdek [1]:

$$J_T = \sum_{k=1}^{P}\sum_{i=1}^{C} \mu_{ik}^2 d_{ik}^2, \tag{9}$$

where d_{ik} is the distance from point k to the center (or prototype) $\vec{h_i}$ of cluster i. We call J_T a typicality measure because it will be small when points in a cluster adhere tightly (have small d_{ik}'s) to their cluster center $\vec{h_i}$.

Density and typicality are important measures because they are closely related to two important characteristics of linguistic terms: the support and the core, respectively. The support is the range of nonzero membership values $(\mu > 0)$, whereas the core is the range of full membership $(\mu = 1)$. In general, we want a linguistic term to have a strong support (high density, or small J_D) and a representative core (good prototype, or small J_T). Thus, it is reasonable to choose $J_D + J_T$ to be our objective function. In other words, for each possible guillotine cut, we calculate $J_D + J_T$ of the resulting partition. Then we select the partition with the least $J_D + J_T$ value as our next hypothesis to continue the hill-climbing

Fig. 8 summarizes our adaptive-network-based fuzzy rule-base modeling scheme. To verify the combined effect of structure identification and parameter identification, we applied the mechanism described above to a time series prediction problem. We generated 400 pieces of training information from a data set containing the monthly mean temperature at Vienna. We used four inputs and 35 rules in this application. After 200 epochs of learning we obtained a modeling error $APE = 4.317\%$.

As mentioned above, the cutting procedure can be viewed as a method of building a decision tree. Efficient decision tree construction depends on the existence of salient discriminating features. If this is not the case, as seen, for example, in Fig. 9, where there are no adequate guillotine cuts, the mechanism of partitioning the feature space is not longer suitable for

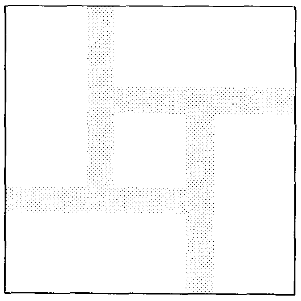


Fig. 9. A clustering without salient discriminating features.

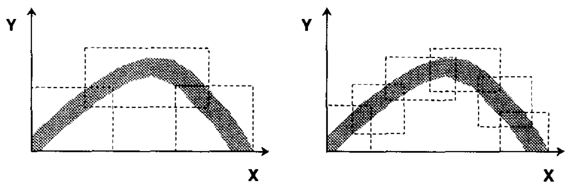


Fig. 10. Fuzzy points approximating a compatibility relation. Various numbers of fuzzy points result in different degrees of information granularity: (a) coarse, (b) finer. The interpretation of fuzzy inference as a compatibility relation was discussed in detail by Ruspini [23].

structure identification. Thus, in very complicated systems, we are usually forced to use large numbers of small rules. In the next section we propose a method of rule organization to cope with the resulting computational complexity.

VII. Rule Organization

In a fuzzy inference system, the rules can also be viewed as a set of fuzzy points which as a whole approximate a compatibility relation. As more rules are involved, finer approximation as well as better modeling accuracy is likely to be achieved (see Fig. 10). However, when modeling accuracy is our major concern and we decide to use an extremely large number of rules, we must deal with the problem of computational complexity.

In this section, we introduce a data structure, called a fuzzy boxtree, to organize rules so that pattern matching can be performed in logarithmic time. The mechanism includes the following steps:

1) use a divide-and conquer data structure, the multilevel fuzzy grid, to partition the feature space and fine-tune a large number of small rules so that accurate local mappings are achieved;
2) define a fuzzy boxtree on antecedents of rules and provide a linear-time algorithm to construct it;
3) introduce a branch-and-bound algorithm for pattern matching in logarithmic time;
4) provide a parallel algorithm to maintain the advantage of parallel processing presumed in fuzzy inference systems;

Since we are going to use a large number of rules to achieve modeling accuracy, and take different degrees of local complexity as well as unbalanced sample distribution into consideration, we adopt a multilevel fuzzy grid [see Fig. 7(d)] as the structure to partition the feature space. The top level grid coarsely partitions the whole space into equal sized and evenly spaced fuzzy boxes, which can be further partitioned by finer fuzzy grids. This partitioning continues until a terminating condition is met. Two criteria can be used as the terminating condition. The first is the balanced sampling criterion; i.e., the resulting boxes should contain similar numbers of training examples. The other is to use an application-dependent evaluation. For example, if we assume each output to be a linear combination of the inputs, as in Jang's model, we can use LMS methods to evaluate the fitness of each grid. When the mean square error is below a threshold, we stop the partitioning process.

Now we can apply the learning model based on adaptive networks to identify the parameters in each region. Because of the small size of the regions and the resulting local linearity, the learning efficiency is expected to be good. Moreover, since the entire feature space is still covered by overlapping regions, the smoothness among regions will not be affected, although the regions are now separately trained. The only problem to be solved is the computational complexity in operation time caused by the resulting large number of rules. Now, we construct a boxtree to put the rules together.

A binary fuzzy boxtree T is a rooted tree in which each internal node has two children. Let R denote the set of nodes of T. Each node $r \in R$ is a fuzzy set with a membership function $\mu_r(u)$ such that

$$\text{if } s \text{ is a child of } r \text{ then } \mu_s(u) \leq \mu_r(u), \forall u \in U,$$
$$\text{in other words, } s \subseteq r.$$

In our application, each leaf stands for a fuzzy pattern which pattern which represents the antecedent part of a certain fuzzy rule. Moreover, each membership function is bellshaped and is determined by six parameters:

$$\mu_A(u) = \begin{cases} \dfrac{1}{1+\left[\left(\dfrac{u-c_1}{a_1}\right)^2\right]^{b_1}}, & u < c_1 \\ 1, & c_1 \leq u \leq c_2 \\ \dfrac{1}{1+\left[\left(\dfrac{u-c_2}{a_2}\right)^2\right]^{b_2}}, & u > c_2, \end{cases} \tag{10}$$

where $c_1 \leq c_2$. Note that the three-parameter function defined in (1) is a special case of this function with $a_1 = a_2, b_1 = b_2$, and $c_1 = c_2$.

The similarity measure between two fuzzy patterns, A and B, is given by the following formula;

$$S(A,B) = \prod_i S(A_i, B_i), \tag{11}$$

i.e., a conjunctive aggregation of partial similarity measures, $S,(A_i, B_i)$'s, in individual feature dimensions. $S(A_i, B_i)$ is in turn calculated by

$$S(A_i, B_i) = \sup_{u \in U} \{\min(\mu_{A_i}(u), \mu_{B_i}(u))\} \tag{12}$$

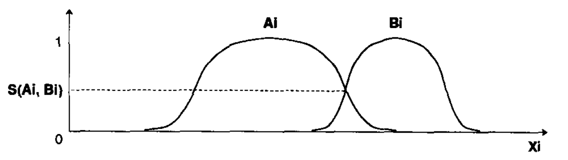


Fig. 11. Similarity measure between two fuzzy sets.

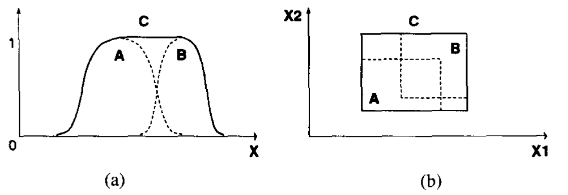


Fig. 12. Constructing an internode in a boxtree: (a) covering of membership functions; (b) corresponding inclusion of boxes.

(see Fig. 11). The boxtree construction algorithm repeatedly finds the two boxes (patterns) with the largest similarity degree, makes them siblings, and inserts the parent as an internode.

A membership function in an internode C is defined as a combination of the corresponding functions in its child nodes, A and B. For example, if A is specified by parameter set $\{a_{1_A}, b_{1_A}, c_{1_A}, a_{2_A}, b_{2_A}, c_{2_A}\}, B$ by $\{a_{1_B}, b_{1_B}, c_{1_B}, a_{2_B}, b_{2_B}, c_{2_B}\}$, and $(c_{1_A}, c_{2_A}) \le (c_{1_B}, c_{2_B})$ in Pareto ordering, we use the following parameters to characterize C:

$$\begin{aligned} a_{1_C} &= a_{1_A} & b_{1_C} &= b_{1_A} & c_{1_C} &= c_{1_A} \\ a_{2_C} &= a_{2_B} & b_{2_C} &= b_{2_B} & c_{2_C} &= c_{2_B}. \end{aligned} \tag{13}$$

Fig. 12 shows the idea and the resulting inclusion relation among fuzzy sets.

Since an internode inherits the (fuzzy) boundaries defined by its children, the above construction can be realized in linear time by employing the famous greedy algorithm of finding a maximum spanning tree, given the similarity measure between each pair of fuzzy patterns. Fig. 13 shows a boxtree constructed.

For $r \in R$ that is a leaf, $\mu_r(u)$ is the compatibility measure of an input u against the pattern r; for an internode $r, \mu_r(u)$ is the upper bound of compatability of the subtree it defines. This property provides a data structure to apply the basic branch-and-bound algorithm in searching optimal solutions. For example, if we want to find all rules which have a firing strength larger than a specified value, the boxtree structure allows a search from the root to prune any subtrees whose root function is smaller than that value.

Thus, we can use the following algorithm to find the best rule against which an input u is matched. In this algorithm, F stands for a frontier of expanded nodes, B is the upper bound to a certain point.

Algorithm 1: *Branch-and-Bound Algorithm to Find the Best-MatchedRule*

```
1) $F \leftarrow \{Root\}; B \leftarrow -\infty$;
2) while $F \neq \emptyset$ do
   select a set of nodes $S \subseteq F$;
   expand the internodes in $S$ to get the set of their children, $L(S)$;
   $F \leftarrow \{F - S\} \cup L(S); B \leftarrow \max(\{B\} \cup \{\mu_v(u) : v \in S \text{ and } v \text{ is a leaf}\})$;
   $F \leftarrow \{v \in F : \mu_v(u) \geq B\}$.   □
```

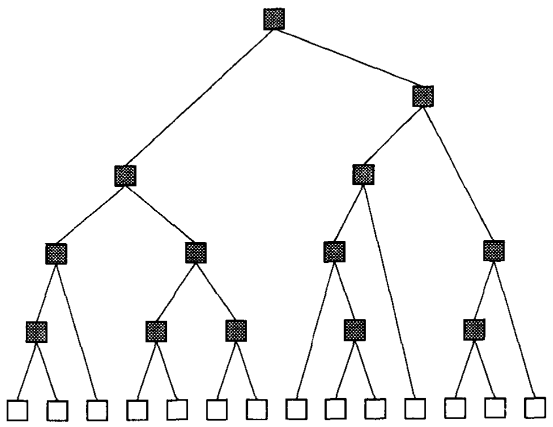


Fig. 13. A boxtree.

Usually, in a fuzzy inference system, it is not necessary to find all rules with a firing strength greater than zero. Instead, we are satisfied with the best k rules whose antecedents are compatible with the input. Algorithm 1 can be generalized to this case by keeping a priority queue of size k and using the kth best value in pruning. This algorithm is of $O(\log_2 R)$ efficiency in pattern marching for a fuzzy rule base with R rules.

The advantage of parallel processing is presumed for fuzzy rule-based inference systems because the rules are considered to be independent of each other in the pattern matching process. Now we organize the rules into a structure, the boxtree; can we still claim the benefit? In other words, if we have p processors instead of one, can we decrease the processing time to $O(1/p)$, or achieve a linear speedup? The answer is positive.

Let each of the p processors maintain a local frontier, F_i, and a local priority queue, B_i. At each step every processor i does one of two things:

1) if $F_i \neq \emptyset$ then it expands the node of best matching in F_i and sends its children to processors chosen at random;
2) if $F_i = \emptyset$ then it sends the message "there is a rule of firing strength s" to processors chosen at random.

The processors then update the sets F_i and queues B_i on the basis of the messages received. The computation continues until all sets F_i are empty. At this point, the best matches are given by the merge of B_i's. This algorithm provides a linear speedup. The details of a general parallel algorithm for a branch-and-bound procedure are given in [34]. The analysis of its complexity is discussed in [20].

In summary, given a fuzzy rule base, with R rules, that models an application system, we can build a boxtree of

$2R-1$ nodes and use a parallel branch-and-bound algorithm to perform the pattern matching task with logarithmic efficiency. Consequently, with the boxtree data structure, we can use many more rules in the modeling process to achieve high performance without losing efficiency in the later pattern matching process.

VIII. Focus-Set- Based Rule Combination

To further improve the performance of self-organized systems, such as those based on adaptive networks, dynamic skeletonization is usually necessary. By skeletonization we mean trimming the redundant or the less important parts of a complicated system, as suggested in [13]. However, we claim that skeletonization should be done under a dynamic relevance criterion, i.e., the part to trim or to simplify should be determined by the current situation. In this paper we propose a method of skeletonization, or rule base compression, for adaptive-network-based fuzzy inference systems.

Note that in Fig. 13 every frontier in a boxtree can be viewed as a fuzzy rule base because it covers the entire feature space. For a frontier containing internodes, we can use either of the following two methods to determine the consequent as well as the antecedent parameters. The first way, the local approach, is to adopt the antecedent parameters as specified in the boxtree and to use LMS methods of finding a hyperplane to approximate the training data covered by individual regions. The alternative, the global approach, is to use the antecedent parameters as the initial values for Jang's model and rerun the entire training process. The latter way is much more time consuming and should be used only when the goal is to find a merged rule base permanently. If we consider the feature space partitioning method introduced previously as a top-down approach, rule merging can be viewed as a bottom-up way of identifying a system structure.

As asserted before, a fuzzy rule-based system can be used to solve various interpolation and classification problems. However, in order to be of practical use for the purpose of system representation and communication, e.g., in medical application of image archiving systems, dynamic rule-base compression becomes essential. With rule compression those components of the system are (temporarily) simplified that are supposed to be of less relevance to the current use of the system. The more relevant a component, the more rules are used for that component. The degree of relevance is specified by a focus set.

A focus set, or a window, is a fuzzy set defined on the feature space which indicates the focus of our current interest. Given a window, W, the similarity gain, $G(r)$, defined on an internode r is calculated as

$$G(r) = S(r_1, W) + S(r_2, W) - S(r, W), \qquad (14)$$

where r_1 and r_2 are the two children of r and S is the similarity measure defined

Now we can use the following algorithm to find the most suitable frontier containing n rules to approximate the original rule base with respect to W.

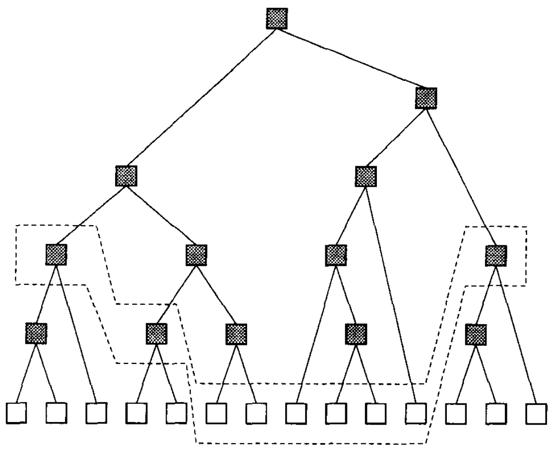


Fig. 14. A compressed rule base. Dotted lines define a frontier in a boxtree. Each frontier can be regarded as a compressed rule-base.

Algorithm 2: *Algorithm to Find the Best-Matched Rule Base withRespect to a Focus Set*

```
1) $F \leftarrow \{Root\}$;
2) while $|F| < n$ do
select an internode $r \in F$ with the largest similarity gain $G(r)$;
expand $r$ to get the set of its children, $L(r)$;
calculate similarity gain for nodes in $L(r)$;
$F \leftarrow \{F - r\} \cup L(r)$; □
```

This algorithm is of linear efficiency.

A compressed rule base with respect to a window is shown in Fig. 14. Once the structure is determined, we apply the local approach mentioned above to identify the consequent parameters. Thus, a simplified but still proper rule base is constructed. It can be used for applications such as coding and hierarchical pattern matching. When higher resolution is required for a simplified region, the corresponding internode can be expanded to provide a finer sub–rule base.

IX. Concluding Remarks

We have proposed a general modeling scheme for an adaptive-network-based fuzzy inference system which can be used in data compression, pattern recognition, decision analysis, and many other fields where human expertise is either unavailable or episodic. The modeling scheme is a two-phase design. Parameter identification is implemented with Jang's model, which employs Kalman filters to improve the overall performance. We modified Jang's model for classification problems. Parameterized t-norms and mean operators were brought into this picture to make the modeling scheme more flexible. We also introduced the concept of weight of importance to achieve the goal of feature selection.

Structure identification, on the other hand, was realized with two different approaches. The top-down method partitions the feature space by guillotine cuts under the guidance of fuzzy clustering objective functions. The two measures, density and typicality, we chose to evaluate clusters have a sound theoretical background in fuzzy sets. The bottom-up approach emphasizes modeling accuracy and uses many small rules. The rules are structured into a fuzzy binary boxtree to speed up the pattern matching process when the rule base is in operation. We proposed a parallel algorithm to maintain the advantage of fuzzy systems in parallel processing.

The rules can also be merged or combined according to a dynamically defined focus set. We provided algorithms of identifying a suitable frontier in a boxtree and determining the parameters. The sub–rule bases thus found can be organized into a hierarchy so that rule bases with various granularities can be employed in accordance with different demands of accuracy and efficiency.

REFERENCES

[1] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York: Plenum Press, 1981.

[2] P. P. Bonissone, "Summarizing and propagating uncertain information with triangular norms," *Int. J. Approximate Reasoning,* vol. 1, pp. 71–101, 1987.

[3] M. A. Eshera and S. C. Barash, "Parallel rule-based fuzzy inference on mesh-connected systolic arrays," *IEEE Expert,* vol. 4, no. 4, pp. 27–35, 1989.

[4] H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Software,* vol. 3, no. 3, pp. 209–226, 1977.

[5] J. S. Jang, "Fuzzy modeling using generalized neural networks and Kalman filter algorithm," in *Proc. 9th National Conf. Artificial Intelligence,* 1991, pp. 762–767.

[6] J. Kacprzyk and R. R. Yager, "'Softer' optimization and control models via fuzzy linguistic quantifiers," *Inform. Sci.,* vol. 34, pp. 157–178, 1984.

[7] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME. J. Basic Engineering,* pp. 35–45, Mar. 1960.

[8] C. Karr, "Applying genetics to fuzzy logic," *AI Expert,* vol. 6, no. 3, pp. 38–43, 1991.

[9] C. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert,* vol. 6, no. 2, pp. 26–33, 1991.

[10] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller," *IEEE Trans. Syst., Man, Cyber.,* vol. 20, no. 2, pp. 404–435, 1990.

[11] M.-H. Lim, "Implementing fuzzy rule-based systems on silicon chips," *IEEE Expert,* vol. 5, no. 1, pp. 31–46, 1990.

[12] J. Moody and C. Darken, "Learning with localized receptive fields," Tech. Rep. YALEU/DCS/RR-649, Dept. Computer Science, Yale University, 1988.

[13] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," Tech. Rep. CU-CS-421-89, Dept. Computer Science and Institute of Cognitive Science, University of Colorado, 1989.

[14] A. Di Nola, W. Pedrycz, S. Sessa, and E. Sanchez, "Fuzzy relation equations theory as a basis of fuzzy modelling: An overview," *Fuzzy Sets and Systems,* vol. 40, pp. 415–429, 1991.

[15] S. M. Omohundro, " Geometric learning algorithms," Tech. Rep. TR-89-041, International Computer Science Institute, 1989.

[16] S. Ovchinnikov, "Similarity relations, fuzzy partitions, and fuzzy ordering," *Fuzzy Sets and Systems,* vol. 40, pp. 107–126, 1991.

[17] S. V. Ovchinnikov and T. Riera, "On fuzzy classifications," in *Fuzzy Set and Possibility Theory,* R. R. Yager, Ed. New York: Pergamon, 1982, pp. 119–132.

[18] W. Pedrycz, "Fuzzy modelling: Fundamentals, construction and evaluation," *Fuzzy Sets and Systems,* vol. 41, pp. 1–15, 1991.

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

[20] A. Ranade, "A simpler analysis of the Karp–Zhang parallel branch-and-bound method," Tech. Rep. UCB/CSD 90/586, Computer Science Division, University of California, Berkeley, 1990.

[21] E. H. Ruspini, "Numerical methods for fuzzy clustering," *Inform Sci.,* vol. 2, pp. 319–350, 1970.

[22] E. H. Ruspini, "Recent development in fuzzy clustering," in *Fuzzy Set and Possibility Theory.* New York: North Holland, 1982, pp. 133–147.

[23] E. H. Ruspini, "On the semantics of fuzzy logic," *Int. J. Approximate Reasoning,* vol. 5, pp. 45–88, 1991.

[24] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems,* vol. 28, pp. 15–33, 1988.

[25] H. Takagi and I. Hayashi, "NN-driven fuzzy reasoning," *Int. J. Approximate Reasoning,* vol. 5, no. 3, pp. 191–212, 1991.

[26] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst. Man, Cyber.,* vol. SMC-15, no. 1, pp. 116–132, 1985.

[27] L.-X. Wang and J. M. Mendel, "Generating fuzzy rules from numerical data, with applications," Tech. Rep. USC-SIPI 169, Signal and Image Processing Institute, University of Southern California, 1990.

[28] R. R. Yager, "Connectives and quantifiers in fuzzy logic," *Fuzzy Sets and Systems,* vol. 40, pp. 39–75, 1991.

[29] R. R. Yager, "Modeling and formulating fuzzy knowledge bases using neural networks," Tech. Rep. MII-1111, Machine Intelligence Institute, Iona College, 1991.

[30] L. A. Zadeh, "Fuzzy sets," *Information and Control,* vol. 8, pp. 338–353, 1965.

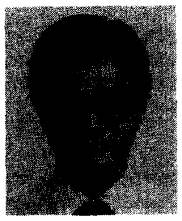
[31] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," *Inform. Sci.,* vol. 8, pp. 199–249, 1975.

[32] L. A. Zadeh, "Fuzzy sets and their application to pattern classification and clustering analysis," in *Classification and Clustering,* J. van Ryzin, Ed. New York: Academic Press, 1978, pp. 251–299.

[33] L. A. Zadeh, unpublished lectures. Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 1991.

[34] Y. Zhang, "Parallel algorithms for combinatorial search problems," Tech. Rep. UCB/CSD 89/543, Computer Science Division, University of California, Berkeley, 1989.

[35] H. J. Zimmermann and P. Zysno, "Latent connectives in human decision making," *Fuzzy Sets and Systems,* vol. 4, pp. 37–51, 1980.



Chuen-Tsai Sun received the B.S. degree in electrical engineering and computer science (1979) and the M.A. degree in history (1984), both from National Taiwan University. He received the Ph.D. degree in computer science from the University of California, Berkeley, in 1992.

He was with the Lawrence Livermore National Laboratory from 1991 to 1992 as a research scientist working on a neuro-fuzzy project. He is currently an Associate Professor with Naional Chiao Tung University, Taiwan. His interests include neuro–fuzzy systems, evolutionary computing, computer vision, and computer assisted instruction.