USC–SIPI REPORT #261

A Fuzzy Classifier That Uses Both Crisp Samples and Linguistic Knowledge

Wen Wei and Jerry M. Mendel

June 1994

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

A Fuzzy Classifier That Uses Both Crisp Samples and Linguistic Knowledge

Wen Wei and Jerry M. Mendel

Signal and Image Processing Institute

Department of Electrical Engineering - Systems

University of Southern California

Los Angeles, CA 90089-2564

June 5, 1994

Abstract

In this paper we develop a general structure of a fuzzy logic (FL) classifier that is capable of using both numerical data and linguistic information. By using fuzzy inference, we are able to handle numerical data and linguistic information in a unified framework. We show that the FL classifier includes the Bayes classifier as a special case. Our experimental results show that the FL classifier, when using linguistic information, can perform better than probabilistic classifiers that do not use linguistic information.

1 Introduction

In many classification applications we want to design classifiers by using numerical prototypes as well as linguistic knowledge. Conventional classification schemes do not provide a systematic method to utilize linguistic knowledge; in contrast, fuzzy logic provides us with a framework for effective management of uncertainty, and is therefore suitable to deal with this problem. Most of the existing supervised fuzzy classification methods are fuzzified versions of crisp classification methods, e.g., the fuzzy perceptron [5] and fuzzy K-nearest-neighbor [6]; hence, they may not lead to substantially different results than their crisp counterparts.

In this paper, we develop a general structure for a classifier that is capable of combining numerical data and linguistic knowledge in a natural manner. Our basic idea is to design a classifier using the principle of fuzzy inference, which has been known to be an effective approximate reasoning technique that deals with vague propositions or statements. By means of fuzzy inference, we are able to handle both numerical data and linguistic knowledge in a unified framework.

In Section 2 we propose a general structure for a Fuzzy Logic (FL) classifier. In Section 3 we investigate the relation of the proposed FL classifier and the Bayes minimum-error classifier. In Section 4 we discuss training methods. Examples are given in Section 5. Conclusions are drawn in Section 6.

2 Constructing Fuzzy Logic Classifiers By Fuzzy Inference

The problem of classification is to categorize a set of objects, which are usually represented by vectors in a feature space. One way to represent a pattern classifier is in terms of a set of discriminant functions, $\{g_i(\mathbf{x}),\ i = 1, 2, \cdots, c\}$, where c is the number of classes (categories), and $\mathbf{x}$ is a feature vector. The classifier assigns $\mathbf{x}$ to class i if $g_i(\mathbf{x}) > g_j(\mathbf{x}), \forall j \neq i$. The feature space is therefore partitioned into c disjoint regions, Γ_1, Γ_2, $\cdots$, Γ_c. These regions can be represented by c characteristic functions defined on the feature space, as follows:

$$\mu_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Gamma_i \\ 0 & \text{otherwise} \end{cases} \quad i = 1, 2, \cdots, c. \tag{1}$$

A classifier can be viewed as a system with input $\mathbf{x}$ and a vector output $\mathbf{y} = col(y_1, y_2, \cdots, y_c)$, where $y_i = \mu_{\Gamma_i}(\mathbf{x})$. A labeled numerical prototype $\mathbf{x}_k$ in class i corresponds to an input-output pair, $(\mathbf{x}_k, \mathbf{y}_k)$, where the i^{th} element of $\mathbf{y}_k$ is 1, and all other elements are 0.

In our scheme of classification, we generalize the $\mu_{\Gamma_i}(\mathbf{x})$'s into fuzzy membership functions. The outputs, γ_i's, of the aforementioned system can then assume any value in [0,1]. A classification problem is thus translated into the problem of approximating the functions describing the system. Our approach to this approximation problem is to use a multi-output Fuzzy Logic System (FLS), since FLS's have been shown [14] to be effective tools to approximate nonlinear functions. Figure 1 shows the configuration of the proposed FL classifier.

2.1 Fuzzy Logic System: A Brief Description

Let U be a universe of discourse. A fuzzy set A over U is characterized by a membership function $\mu_A(u)$ with $u \in U$. A linguistic variable X takes a fuzzy set as its value.

The FLS used in our FL classifier consists of four parts: a fuzzifier, a fuzzy inference engine, a fuzzy rule base, and a defuzzifier. We assume singleton fuzzification is used. The fuzzy rule base consists of a set of *IF-THEN* rules in the form of:

$$\textit{IF } X \textit{ is } A, \textit{ THEN } Y \textit{ is } B, \tag{2}$$

where X and Y are linguistic variables, and A and B are fuzzy sets over universe of discourses U and V, respectively. A typical form of fuzzy inference is:

$$\begin{array}{ll} \textit{Premise 1} & X \textit{ is } A' \\ \textit{Premise 2} & \textit{If } X \textit{ is } A, \textit{ Then } Y \textit{ is } B \\ \textit{Conclusion} & Y \textit{ is } B'. \end{array} \tag{3}$$

In this way, the inference engine matches an input with a fuzzy rule to find the conclusion B'. The compositional rule of fuzzy inference [15] states that

$$\mu_{B'}(v) = \sup_{u \in U}\{\mu_{A'}(u) \star \mu_{(A \to B)}(u, v)\}, \tag{4}$$

where $\star$ is an arbitrary T-norm [9] (e.g., minimum or product), and $(A \to B)$ is a fuzzy implication.

There are many ways to define a fuzzy implication [9]. In FLS's, T-norm implication is often used, i.e.,

$$\mu_{(A\rightarrow B)}(u,v) = \mu_A(u) \star \mu_B(v) \tag{5}$$

Usually we deal with FLS's with vector input and scalar output. For notational clarity we use x and y, instead of u and v, to represent the input and output variables. In the fuzzy logic system's literature, (e.g., [12]), multi-antecedent fuzzy rules are often used, such as:

$$\begin{gathered} \textit{If } X_1 \textit{ is } A_1, \textit{ and } X_2 \textit{ is } A_2, \cdots, \textit{ and } X_d \textit{ is } A_d, \\ \textit{Then } Y \textit{ is } B, \end{gathered} \tag{6}$$

where d is the dimension of x. T-norms are used to combine the antecedents:

$$\mu_{(A_1,A_2,\ldots,A_d)}(x_1,x_2,\ldots,x_d) = \mu_{A_1}(x_1) \star \mu_{A_2}(x_2) \star \cdots \star \mu_{A_d}(x_d) \tag{7}$$

Note that we can use the rule in (2) to represent the multi-antecedent rule, by defining the membership function of A as

$$\mu_A(\mathrm{x}) = \mu_{(A_1,A_2,\ldots,A_d)}(x_1,x_2,\ldots,x_d). \tag{8}$$

When there are N *IF-THEN* rules in the fuzzy rule base, such as

$$\textit{If } X \textit{ is } A^i, \textit{ Then } Y \textit{ is } B^i, \; i = 1,2,\cdots,N, \tag{9}$$

we will obtain N inferred fuzzy sets from fuzzy inference. Let B'^i denote the inferred fuzzy set from the i^{th} rule. A T-conorm, denoted by $\oplus$, (e.g., maximum or bounded sum) is often used to combine the B'^i's to get the overall output fuzzy set, B', i.e.,

$$B' = B'^1 \oplus B'^2 \oplus \cdots \oplus B'^N. \tag{10}$$

If A' is a singleton at x' (as in singleton fuzzification), i.e., $\mu_{A'}(\mathrm{x})$ is unity at x' and zero elsewhere, then (4) becomes

$$\mu_{B'}(y) = 1 \star \mu_{(A \to B)}(\mathrm{x}', y) = \mu_A(\mathrm{x}') \star \mu_B(y) \tag{11}$$

The fuzzy inference engine and fuzzy rule base described above determine a mapping from a point x' to a fuzzy set B'. The defuzzifier maps B' to a crisp value, $\bar{y}$. In our FL classifier, we use the Centroid of Area (COA) defuzzifier [9], i.e.,

$$\bar{y} = \frac{\sum \mu_{B'}(y) y}{\sum \mu_{B'}(y)}, \tag{12}$$

where the summation is over the universe of discourse, and, if the universe of discourse is continuous, the summation is replaced with an integral.

For a multi-output FLS with n outputs, fuzzy rules, such as

$$\begin{gathered} \textit{If } \mathrm{x} \textit{ is } A, \\ \textit{Then } y_1 \textit{ is } B_1,\ y_2 \textit{ is } B_2,\ \cdots, \textit{ and } y_n \textit{ is } B_n, \end{gathered} \tag{13}$$

have multiple consequents, and we can decompose them into the following n single-consequent rules: *If* x *is* A, *Then* y_j *is* B_j, $j = 1, 2, \cdots, n$. In this way, we can treat the FLS as n parallel single-output FLS's.

2.2 Representing Linguistic Knowledge By Fuzzy Rules

Fuzzy rules are suitable to represent linguistic knowledge [16]. In order to use an FLS to approximate the functions, $y_i = \mu_{\Gamma_i}(\mathrm{x})$, we need fuzzy rules in the following form:

$$\textit{IF } X \textit{ is } A, \textit{ THEN } Y_i \textit{ is } B_i,\ i = 1, 2, \cdots, c, \tag{14}$$

where X is a linguistic variable that represents a feature of the object, Y_i is a linguistic variable that represents the approximate value of $\mu_{\Gamma_i}(\mathrm{x})$ (i.e., the membership grade of the object in class i), A is a fuzzy set on the feature space, and B_i's are *fuzzy numbers* on [0,1]. We use an example to illustrate the meaning of A and B_i. Consider the problem of classifying grape fruits (S_1) and oranges (S_2) by comparing their radii (R). We know that *grape fruits are bigger than oranges.* This knowledge is converted into the following fuzzy *IF-THEN* rule: "*IF R is BIG, THEN Y_1 is LARGE and Y_2 is SMALL,*" where R represents the approximate value of the radius of the object, Y_1 and Y_2 represent the similarity of the object to grape fruits and oranges respectively, *BIG* is a fuzzy set in feature space (see Figure 2a), and, *LARGE* and *SMALL* are fuzzy sets in output space (see Figure 2b). In our FL classifier, we assume that the following L linguistic rules are available:

$$\begin{gathered} \textit{IF}\ \mathrm{x}\ \textit{is}\ A^j, \\ \textit{THEN}\ y_1\ \textit{is}\ B_1^j,\ \textit{and}\ \cdots,\ \textit{and}\ y_c\ \textit{is}\ B_c^j, \end{gathered} \tag{15}$$

for $j = 1, 2, \cdots, L$.

2.3 Extracting Fuzzy Rules From Numerical Data

Here we describe two methods for extracting fuzzy rules from numerical data.

A) **Direct Method**

Let $\{\mathrm{x}_k^{(i)}, k = 1, 2, \cdots, N_i\}$ be a set of prototypes for class i, $i = 1, 2, \cdots, c$. Our principle of rule extraction is that a point x near $\mathrm{x}_k^{(i)}$ probably belongs to the same class, numbered i. We first construct a fuzzy set, $F(k|i)$, to represent the "neighborhood" of $\mathrm{x}_k^{(i)}$. Naturally, $\mu_{F(k|i)}(\mathrm{x})$ should peak at $\mathrm{x} = \mathrm{x}_k^{(i)}$, and decrease as some distance measure between x and $\mathrm{x}_k^{(i)}$ increases. Let us use the Mahalanobis distance with positive definite matrix Q_i for class i; then, $\mu_{F(k|i)}(\mathrm{x})$ can be written as

$$\mu_{F(k|i)}(\mathrm{x}) = \phi(\|\mathrm{x} - \mathrm{x}_k^{(i)}\|_{Q_i}), \tag{16}$$

in which,

$$\|\mathbf{x} - \mathbf{x}_k^{(i)}\|_{Q_i} = \sqrt{(\mathbf{x} - \mathbf{x}_k^{(i)})^t Q_i^{-1} (\mathbf{x} - \mathbf{x}_k^{(i)})} \qquad (17)$$

and $\phi(\bullet)$ is a function that satisfies: (1) $\phi(0) = 1$, and (2) $\phi(x_1) \geq \phi(x_2)$ if $|x_1| < |x_2|$. A direct rule-extraction method is to generate a fuzzy rule (hereafter called a numerical rule) from each prototype in the following way:

$$\textit{IF } X \textit{ is } F(k|i), \textit{ THEN } Y_i \textit{ is 1 and } Y_j \textit{ is 0}, \ j \neq i, \ i = 1, 2, \cdots, c, \qquad (18)$$

in which fuzzy singletons (0 or 1) are used in the consequent parts, since we know that $\mathbf{x}_k^{(i)}$ belongs to class i with degree 1. This rule can be read as: if the object is close to $\mathbf{x}_k^{(i)}$, then it belongs to class i.

The number of rules by the Direct Method is equal to the total number of prototypes; hence, this method is suitable for small prototype sets.

2) Cluster Analysis

An alternative to the Direct Method is to divide the prototypes into a small number of groups, and generate a rule for each group. Cluster analysis is a way to group the prototypes. There are a variety of clustering techniques that can be used for this purpose, e.g., the k-means family (fuzzy or non-fuzzy) of clustering algorithms [1], and neural network classifiers [8].

Suppose M_i clusters are obtained for class i. Compute the centroid of each cluster, denoted by $\mathbf{z}_k^{(i)}$. Let $G(k|i)$ be a fuzzy set induced by $\mathbf{z}_k^{(i)}$, with membership function:

$$\begin{aligned} &\mu_{G(k|i)}(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{z}_k^{(i)}\|_{Q_i}), \\ &k = 1, 2, \cdots, M_i, \ i = 1, 2, \cdots, c, \end{aligned} \qquad (19)$$

We can then create the following fuzzy rules:

$$\textit{IF } X \textit{ is } G(k|i), \textit{ THEN } Y_i \textit{ is 1 and } Y_j \textit{ is 0}, \ j \neq i, \ i = 1, 2, \cdots, c. \qquad (20)$$

We can view the Direct Method as a special case of cluster analysis methods, in which a cluster is created for each prototype; hence, (20) is a general form for numerical rules.

2.4 A Fuzzy Classifier Using Fuzzy Inference

We can now construct a classifier that uses the linguistic rules in (15) and the numerical rules in (20). Denote by L_i^j the inferred fuzzy set for Y_i from linguistic rule j, and by $N_j(k|i)$ the inferred fuzzy set for Y_j from the numerical rule associated with $\mathbf{z}_k^{(i)}$. Using (15) and (11), we have

$$\begin{aligned}&\mu_{L_i^j}(y) = \mu_{A^j}(\mathbf{x}) \star \mu_{B_i^j}(y),\\ &i = 1, \cdots, c,\ j = 1, 2, \cdots, L,\end{aligned} \tag{21}$$

and, using (19), (20), and (11), we have

$$\begin{aligned}&\mu_{N_j(k|i)}(y) = \begin{cases} \phi(\|\mathbf{x} - \mathbf{z}_k^{(i)}\|_{Q_i})\delta(y-1) & \text{if } j = i, \\ \phi(\|\mathbf{x} - \mathbf{z}_k^{(i)}\|_{Q_i})\delta(y) & \text{if } j \neq i \end{cases}\\ &i, j = 1, \cdots, c,\ k = 1, \cdots, M_i,\end{aligned} \tag{22}$$

where $\delta()$ is the Kronecker delta function. Note that we have used the following fact: $\phi(\bullet) \star \delta(\bullet) = \phi(\bullet)\delta(\bullet)$, since $\delta(\bullet)$ takes only 0 and 1 as its values.

Now we can apply (10) to combine the inferred results; however, we usually have different belief factors for linguistic and numerical rules; therefore, we assign a weight factor to each fuzzy rule (α_j for a linguistic rule and β_k^i for a numerical rule), and use (10), (21), and (22) to conclude that, for $i = 1, 2, \cdots, c$,

$$\begin{aligned}\mu_{B_i'}(y) = &\left(\bigoplus_{j=1}^{L} \alpha_j(\mu_{A^j}(\mathbf{x}) \star \mu_{B_i^j}(y))\right) \oplus \left(\bigoplus_{k=1}^{M_i} \beta_k^i \phi(\|\mathbf{x} - \mathbf{z}_k^{(i)}\|_{Q_i})\delta(y-1)\right) \oplus \\ &\left(\bigoplus_{j=1, j\neq i}^{c} \bigoplus_{k=1}^{M_j} \beta_k^j \phi(\|\mathbf{x} - \mathbf{z}_k^{(i)}\|_{Q_i})\delta(y)\right)\end{aligned} \tag{23}$$

where the big $\bigoplus$ represents L (or M_i, or c) cascaded $\oplus$ operations. Figure 3 illustrates (23) for $c = 5$ triangular membership functions for $\mu_{B_i'}(y)$, and minimum inference for the linguistic rules. The trapezoids correspond to the first term in (23), the spikes at $y = 1$ correspond to the second term, and the spikes at $y = 0$ correspond to the third term. Note that, prior to taking all the T-conorms in the third term, there are many constituents to that term; however, after taking these T-conorms, only one survives. Note, also, that our example has assumed no overlap between the trapezoidal membership functions and those at $y = 0$ and $y = 1$.

We can now defuzzify B_i' to obtain the value of $\mu_{\Gamma_i}(\mathrm{x})$, i.e., $\mu_{\Gamma_i}(\mathrm{x}) = \bar{y}_i = COA(B_i')$, $i = 1, 2, \cdots, c$. Note that $\mu_{\Gamma_i}(\mathrm{x})$ provides the degree of similarity of x to class i. If a crisp decision is needed, we can compare all $\mu_{\Gamma_i}(\mathrm{x})$'s over i, and assign x to the class with maximum membership. For example, in Figure 3, the y coordinate of the "x" points represents the COA values of B_i'. Since $COA(B_3')$ is the maximum, our decision is class 3.

Equation (23) represents a family of classifiers capable of using both crisp numerical data and linguistic knowledge. By selecting a pair of specific T-norms and T-conorms, we can reach various kinds of classifiers. The parameters, α_j's, β_k^i's, and Q_i, can be determined by trial-and-error or by using some training algorithms.

3 Relation between the Additive FL Classifiers and the Bayes Classifiers

Combining rules additively [7] is a technique that uses addition in place of a T-conorm to combine the inferred fuzzy sets from fuzzy rules. A FL classifier that combines rules additively is referred to as an Additive Fuzzy Logic Classifier (AFLC). Obviously, the value of $\mu_{B_i'}(y)$ for an AFLC can be greater than unity; however, this problem is easily handled by our FL classifier because we can scale $\mu_{B_i'}(y)$ by a common factor without changing the defuzzified value of B_i' (see (12)). In this section we will study the relation between AFLCs and Bayes classifiers.

Fuzzy and probabilistic classifiers are not always related to each other because probabilistic quantities may not exist for some classification problems. In order to investigate their relation, we suppose that there exist underlying probability quantities, i.e., an a priori probability, P_i, for each class, and, class-conditional joint probability densities, $p_i(\mathrm{x})$, $i = 1, 2, \cdots, c$.

3.1 Relating the Two Classifiers by using Special Linguistic Rules

Theorem 1: The AFLC gives the same crisp classification results as the Bayes classifier, if: (1) all $p_i(\mathrm{x})$'s are bounded by a number $p_{\max}$, (2) c linguistic rules are used in the FL classifier, with $\mu_{A^j}(\mathrm{x}) = p_j(\mathrm{x})/p_{\max}$, $\alpha_j = P_j$, and $\mu_{B_i^j}(y) = \delta(i-j)\delta(y-1) + (1-\delta(i-j))\delta(y)$, $i, j = 1, 2, \cdots, c$, and (3) no numerical rules are used in the FL classifier.

Proof: Using conditions (1)-(3) in (23), we have

$$\begin{aligned}\mu_{B_i'}(y) &= \oplus_{j=1}^{c} P_j((p_j(\mathrm{x})/p_{\max}) \star \\ &\quad (\delta(i-j)\delta(y-1) + (1-\delta(i-j))\delta(y))) \\ &= \begin{cases} P_i p_i(\mathrm{x})/p_{\max} & \text{if } y = 1 \\ \sum_{j=1, j\neq i}^{c} P_j p_j(\mathrm{x})/p_{\max} & \text{if } y = 0 \\ 0 & \text{otherwise} \end{cases}\end{aligned} \tag{24}$$

Using COA defuzzification (12), we obtain

$$\mu_{\Gamma_i}(\mathrm{x}) = COA(B_i') = \frac{P_i p_i(\mathrm{x})}{\sum_{j=1}^{c} P_j p_j(\mathrm{x})} \tag{25}$$

On the other hand, the discriminant functions of the Bayes classifier are [3]:

$$g_i(\mathrm{x}) = P_i p_i(\mathrm{x}), \; i = 1, 2, \cdots, c. \tag{26}$$

Since $g_i(\mathrm{x})$ equals $\mu_{\Gamma_i}(\mathrm{x})$ multiplied by a term that does not depend on i, the FL classifier gives the same crisp decisions as the Bayes classifier. □

3.2 Relating the Two Classifiers by using Special Numerical Rules

The probabilistic classifier we consider here is the Bayes Minimum Error Classifier using Kernel Estimation (BMECKE), which uses the following discriminant functions [3]:

$$g_i(\mathbf{x}) = \frac{P_i}{N_i \sigma_i^d} \sum_{k=1}^{N_i} \mathcal{K}\left(\|\mathbf{x} - \mathbf{x}_k^{(i)}\|/\sigma_i\right) \tag{27}$$

where d is the dimension of $\mathbf{x}$, $\|\bullet\|$ is the Euclidean distance, and $\mathcal{K}()$ is a kernel function. The reasons why we consider the BMECKE are: (1) kernel estimations converge to the true probability densities under a broad range of conditions, and thus the BMECKE is a good approximation of the optimal Bayes classifier; and (2) the BMECKE has been well studied [3, 11].

Theorem 2: An AFLC gives the same crisp classification results as the BMECKE, if: (1) The Direct Method (Section 2.2) is used to generate numerical rules; (2) no linguistic information is used; and, (3) $\mathcal{K}(\mathbf{x})$ is bounded by $\mathcal{K}_{\max}$, $\phi(\bullet) = \mathcal{K}(\bullet)/\mathcal{K}_{\max}$, $\beta_k^i = P_i/N_i$, and, $Q_i = \sigma_i^2 I$.

Proof: When using the Direct Method to generate numerical rules, $\{\mathbf{z}_k^{(i)}\}$ in (23) are the same as $\{\mathbf{x}_k^{(i)}\}$. From conditions (2) and (3), and using (23), we have, for $i = 1, \cdots, c$,

$$\mu_{B_i'}(y) = \begin{cases} \frac{P_i}{N_i \mathcal{K}_{\max}} \sum_{k=1}^{N_i} \mathcal{K}(\|\mathbf{x} - \mathbf{x}_k^{(i)}\|/\sigma_i) & y = 1 \\ \sum_{j=1, j \neq i}^{c} \frac{P_j}{N_j \mathcal{K}_{\max}} \sum_{k=1}^{N_j} \mathcal{K}(\|\mathbf{x} - \mathbf{x}_k^{(j)}\|/\sigma_i) & y = 0 \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

Hence, COA defuzzification gives

$$\mu_{\Gamma_i}(\mathbf{x}) = \frac{\frac{P_i}{N_i} \sum_{k=1}^{N_i} \mathcal{K}(\|\mathbf{x} - \mathbf{x}_k^{(i)}\|/\sigma_i)}{\sum_{j=1}^{c} \sum_{k=1}^{N_j} \frac{P_j}{N_j} \mathcal{K}(\|\mathbf{x} - \mathbf{x}_k^{(j)}\|/\sigma_i)} \tag{29}$$

It is easy to see that $\mu_{\Gamma_i}(\mathbf{x})$ differs from $g_i(\mathbf{x})$ in (27) only by a term that does not depend on i; thus, the two classifiers give the same classification results. □

3.3 Discussions

Theorem 1 says that if we use the class-conditioned probability densities for antecedents of fuzzy rules, and the a priori probabilities as the weighting factors, the AFLC reduces to the Bayesian classifier. Consider the example of classifying grape fruits and oranges, in which the a priori probabilities of grape fruits and oranges are P_g and P_o, the probability densities of their radii are $p_g(r)$ and $p_o(r)$, and the latter are bounded by $p_{\max}$. Let fuzzy sets A^1 and A^2 be: $\mu_{A^1}(r) = p_g(r)/p_{\max}$, and $\mu_{A^2}(r) = p_o(r)/p_{\max}$. The linguistic rules that correspond to those in Theorem 1 are: (1) *IF R is A^1, THEN Y_1 is 1, Y_2 is 0,* and, (2) *IF R is A^2, THEN Y_2 is 1, Y_1 is 0,* which mean that if R is A^1, then it is a grape fruit, and if R is A^2, then it is an orange.

In Theorem 2, the only probabilistic information used in the AFLC is the a priori probabilities, the P_i's. If the P_i's are unknown, a natural choice is to set $\beta^i_k = 1/N_i$, since by doing so, all classes are treated equally. In addition, assuming equal a priori probabilities is also a natural choice for the BMECKE. In this case, the AFLC and BMECKE give the same result; hence, we can reach the same classifier as the BMECKE ***without using any probability concepts.***

From this we see that there is an intrinsic relation between combining rules additively and probability principles. In light of this relation, we speculate that additive combining is suitable for independently drawn prototypes, as is kernel estimation. On the other hand, if the prototypes are dependent, or, if the rules are generated by cluster analysis, then the rules will not be independent, and therefore other operators should be used to combine rules.

4 Parameter Training for AFLC

Parameter training selects a set of parameters for the FL classifier that optimize some predefined criterion function. For a classifier, the number of misclassifications is an obvious criterion. When the underlying probability quantities are all known, the Bayesian classifier is optimal; therefore, the parameters given in Theorem 1 are optimal for the FL classifier. On the other hand, when we know only a set of samples, we can not evaluate the number of misclassifications; instead, we use an error count within the sample set as a criterion.

In the rest of this paper, we will consider only a Gaussian function for $\phi()$, i.e., $\phi(x) = \exp\{-x^2/2\}$. In the FL classifier, the following four sets of parameters must be determined: α's, β's, Q's, and z's. The number of parameters in each parameter set is: $N(\alpha) = L$, $N(\beta) = N(\mathbf{z}) = \sum_{i=1}^{c} M_i$, and $N(Q) = cd^2$ (where $d = dim(\mathbf{x})$). α's represent the belief measure of the linguistic rules, so they should be determined in the rule-extraction procedure. If human experts provide linguistic rules, they may also be able to provide belief factors for the rules. Here we only discuss some methods to determine the other parameters from training samples.

4.1 Determining $\mathbf{z}_k^{(i)}$

As mentioned in Section 2.3, cluster analysis methods can be used to initialize the $\mathbf{z}_k^{(i)}$'s. The following Fuzzy c-Means algorithm (FCM) has been widely used in rule generation for FLS's.

FCM Problem [1]: Suppose we have a set of vectors, $\mathbf{x}_p, p = 1, 2, \cdots, N$, and we wish to group the data into M clusters. The FCM problem is to find M *cluster centers*, $V = [\mathbf{v}_k], k = 1, \cdots, M$, and a *fuzzy partition* matrix, $U = [u_{kp}], k = 1, \cdots, M, p = 1, \cdots, N$, by minimizing the following criterion function:

$$J_m(U, V) = \sum_{p=1}^{N} \sum_{k=1}^{M} u_{kp}^m \|\mathbf{x}_p - \mathbf{v}_k\|^2, \tag{30}$$

where $m > 1$ is a parameter, usually chosen to be 2. u_{kp} can be viewed as the similarity between $\mathbf{x}_p$ and cluster k. When u_{kp} assumes only binary values, i.e., $u_{kp} = 1$ or 0, (meaning that $\mathbf{x}_k$ does or does not belong to cluster p, respectively), then $J_m(U, V)$ is the total in-cluster squared error. In the FCM problem, $u_{kp} \in [0, 1]$, which means that $\mathbf{x}_p$ can belong to more than one cluster to different degree of similarity. The solution to this clustering problem is the following:

FCM Algorithm: Iteratively, perform the following computations:

$$\mathbf{v}_k^{(l+1)} = \frac{\sum_{p=1}^{N} (u_{kp}^{(l)})^m \mathbf{x}_p}{\sum_{p=1}^{N} (u_{kp}^{(l)})^m}, \quad k = 1, \cdots, M \tag{31}$$

and,

$$u_{kp}^{(l+1)} = \begin{cases} \frac{1}{\sum_{j=1}^{M}(d_{kp}^{(l+1)}/d_{jp}^{(l+1)})^{1/(m-1)}} & \text{if } d_{jp}^{(l+1)} \neq 0,\ j = 1, \cdots, M \\ 1 & \text{if } d_{kp}^{(l+1)} = 0 \\ 0 & \text{if } d_{jp}^{(l+1)} = 0,\ j \neq k \end{cases} \tag{32}$$

where $k = 1, \cdots, M$, and,

$$d_{jp}^{(l+1)} = \|\mathbf{x}_p - \mathbf{v}_j^{(l+1)}\|^2, \quad \forall j = 1, \cdots, M \text{ and } p = 1, \cdots, N \tag{33}$$

It has been proved that the FCM algorithm always converges [2]. In practice, the algorithm is initialized with a randomly picked $u_{kp}^{(0)}$ (or $\mathbf{v}_p^{(0)}$), and is stopped when $u_{kp}^{(l+1)} - u_{kp}^{(l)}$ (or $\| \mathbf{v}_p^{(l+1)} - \mathbf{v}_p^{(l)} \|$) is smaller than a threshold.

When the FCM algorithm is used for the samples in each class (and thus, c sets of u_{kp} and $\mathbf{v}_p$ are obtained), the resulting cluster centers provide representative vectors in the class, and thus are used as the initial values of our $\mathbf{z}_k^{(i)}$'s. The number of clusters, M_i, is determined based on the number of samples in class i, and the constraints on the maximum number of rules. Figure 4 depicts 50 points from two classes, and 4 clusters centers found by FCM for each class.

4.2 Determining β_k^i

The number of samples that are close to a $\mathbf{z}_k^{(i)}$ reflects the "typicalness" of the $\mathbf{z}_k^{(i)}$. It is reasonable to assign a large weighting factor, β_k^i, to a $\mathbf{z}_k^{(i)}$ that has a large concentration of samples around it. Suppose M_i cluster centers are obtained for class i. Since c sets of u_{kp} are obtained for the c classes, in order to distinguish among them, we add a superscript to u_{kp}; i.e., we denote by u_{kp}^i the convergent value of $u_{kp}^{(l+1)}$ for class i. We use the following ad hoc measure of concentration for β_k^i:

$$\beta_k^i = \sum_{p=1}^{N_i} (u_{kp}^i)^2, \ k = 1, \cdots, M_i, \ i = 1, \cdots, c. \tag{34}$$

4.3 Determining Q_i

We consider only a special case when $Q_i = \sigma_i^2 I$. How to choose the σ_i's is also a problem in the design of BMECKE. According to Specht [11], BMECKE tends to a nearest-neighbor classifier as $\sigma_i \to 0$, and it tends to a linear classifier as $\sigma_i \to \infty$. Specht reports that it is not difficult to find a good value for σ_i by trial-and-error, because the misclassification rate does not change drastically with σ_i. The initial value of σ_i can be computed from the average distance between all pairs of samples, i.e.,

$$\sigma_i^{(0)} = \frac{\sum_{k=1}^{N_i} \sum_{p=1,p\neq k}^{N_i} d(\mathbf{x}_k, \mathbf{x}_p)}{N_i(N_i - 1)}. \tag{35}$$

It has been shown [10] that kernel estimation is the smoothest if $\sigma_i^{(0)}$ is proportional to $N_i^{-0.2}$. We found that when N_i is very large $\sigma_i^{(0)}$ tends to a constant; hence, we compute σ_i as:

$$\sigma_i = \frac{\sigma_i^{(0)}}{N_i^{0.2}} \tag{36}$$

where $\sigma_i^{(0)}$ is given by (35).

4.4 Training σ_i for Two-Class AFLC

A trial-and-error method, which computes the misclassification count for each value of σ_i, needs a great deal of computation when the number of samples is large. Here we propose a gradient-search method to compute σ_i.

When there are no linguistic rules, then, from (23) and (12), we have [$\mu_i(\mathbf{x})$ stands for $\mu_{\Gamma_i}(\mathbf{x})$]

$$\mu_i(\mathbf{x}) = COA(B_i') = \sum_{k=1}^{M_i} \beta_k^i \phi(\|\mathbf{x} - \mathbf{z}_k^{(i)}\|/\sigma_i)/C(\mathbf{x}, \sigma_1, \sigma_2) \tag{37}$$

where $C(\mathbf{x}, \sigma_1, \sigma_2)$ does not depend on i, so it cancels when we compare μ_1 and μ_2, i.e., $C(\mathbf{x}, \sigma_1, \sigma_2)$ does not affect the classification result; hence, in the following we drop $C(\mathbf{x}, \sigma_1, \sigma_2)$ from $\mu_i(\mathbf{x})$. Because we want to view σ_i as a variable, we denote $\mu_i(\mathbf{x})$ as $\mu_i(\mathbf{x}, \sigma_i)$.

Our ultimate purpose is to minimize the misclassification count. For the two-class case,

$\mu_1(\mathrm{x}_k^{(1)}, \sigma_1) > \mu_2(\mathrm{x}_k^{(1)}, \sigma_2)$ implies a correct classification of $\mathrm{x}_k^{(1)}$; hence, the count of correct classification can be written as

$$\mathcal{N}(\sigma_1, \sigma_2) = \sum_{k=1}^{N_1} u[\mu_1(\mathrm{x}_k^{(1)}, \sigma_1) - \mu_2(\mathrm{x}_k^{(1)}, \sigma_2)] + \sum_{k=1}^{N_2} u[\mu_2(\mathrm{x}_k^{(2)}, \sigma_2) - \mu_1(\mathrm{x}_k^{(2)}, \sigma_1)] \tag{38}$$

where $u()$ is the unit step function. Our goal is to maximize $\mathcal{N}(\sigma_1, \sigma_2)$. Unfortunately, this count is not a differentiable function of σ_i; hence, we use a sigmoidal function, $f(x) = 1/(1 + e^{-\gamma x})$ to approximate the step function, where γ is a fixed positive number. In this way, we get the following criterion function:

$$\mathcal{N}(\sigma_1, \sigma_2) \approx J(\sigma_1, \sigma_2) = \sum_{k=1}^{N_1} f[\mu_1(\mathrm{x}_k^{(1)}, \sigma_1) - \mu_2(\mathrm{x}_k^{(1)}, \sigma_2)] + \sum_{k=1}^{N_2} f[\mu_2(\mathrm{x}_k^{(2)}, \sigma_2) - \mu_1(\mathrm{x}_k^{(2)}, \sigma_1)] \tag{39}$$

Our gradient-search algorithm for updating σ_i is:

$$\sigma_1^{(l+1)} = \sigma_1^{(l)} + \delta^{(l)} \frac{\partial J}{\partial \sigma_1} \tag{40}$$

$$\sigma_2^{(l+1)} = \sigma_2^{(l)} + \delta^{(l)} \frac{\partial J}{\partial \sigma_2} \tag{41}$$

This is a many-at-a-time algorithm, i.e., it needs to compute $N_1 + N_2$ derivatives at every iteration. As in many other hill-climbing algorithms, this algorithm can also be implemented in a one-at-a-time manner. The trick is to use *one pair* of samples at a time; specifically, to replace $J(\sigma_1, \sigma_2)$ in (40) and (41) by

$$J_k(\sigma_1, \sigma_2) = f[\mu_1(\mathrm{x}_k^{(1)}, \sigma_1) - \mu_2(\mathrm{x}_k^{(1)}, \sigma_2)] + f[\mu_2(\mathrm{x}_k^{(2)}, \sigma_2) - \mu_1(\mathrm{x}_k^{(2)}, \sigma_1)], \tag{42}$$

and to increment k in each iteration, i.e., to use the training samples iteratively.

Figure 5a shows the performance surface of $J(\sigma_1, \sigma_2)$ (with $\gamma = 10$) for the samples and cluster centers depicted in Figure 4. Observe that $J(\sigma_1, \sigma_2)$ has a very small gradient on a plateau of maximum values, which makes the gradient-search algorithm converge very slowly. To overcome this problem, we fixed σ_2 at its initial value, [initial values of σ_i are computed by (36)], and applied our training algorithm only for σ_1. This results in a point on the plateau, which is okay because $J(\sigma_1, \sigma_2)$ is close to its maximum value at any point on the plateau. Figure 5b shows the performance surface of $J(\sigma_1, \sigma_2)$ for several values of σ_2. Observe that $J(\sigma_1, \sigma_2)$ is a concave function of σ_1, which assures fast convergence of the gradient-search algorithm. Observe, also, that a global maximum of $J(\sigma_1, \sigma_2)$ occurs when $\sigma_1 \approx 2.2$ and $\sigma_2 \approx 4$.

5 Examples and Discussions

Example 1: Following Ishibuchi et al. [4], we designed a two-class classifier on a pattern space $[0, 20] \times [0, 20]$. The numerical data are: $S_1 = \{(4, 11), (8, 11), (11, 3), (13, 4), (13, 10)\}$ and $S_2 = \{(2, 13), (6, 14), (13, 2), (14, 3), (14, 14)\}$. The following two linguistic rules were used: (1) *IF x_1 is SMALL and x_2 is SMALL, THEN y_1 is 1 and y_2 is 0,* and (2) *IF x_1 is VERY LARGE or x_2 is VERY LARGE, THEN y_2 is 1 and y_1 is 0.* The membership functions of fuzzy sets SMALL and VERY LARGE are shown in Figure 6.

Classifier Design: We used the AFLC with product inference, because no information was known about the dependency of the data. In generating numerical rules we used the Direct Method and Gaussian membership functions, i.e., $\phi(x) = \exp\{-x^2/2\}$. The α and β were all set to unity. Because the number of training samples is small, we did not use (36) to compute σ_i; instead, we picked two values, $\sigma_1 = \sigma_2 = 3$ and $\sigma_1 = \sigma_2 = 5$.

Results: Figure 7 shows the decision boundaries of the BMECKE using only numerical data. Figure 8 shows the decision boundaries of the AFLC using both numerical data and linguistic information with the same values for σ_i. We see that: (1) both classifiers classify all prototypes correctly, and (2) the fuzzy classifier correctly conveys the effects of the linguistic rules. These results are comparable to the neural network classifier in [4], which is trained by the back-propagation algorithm; however, the AFLC is nearly a one-pass process, whereas the neural network classifier may need a significant amount of time for training.

Example 2: The data were randomly generated from two exponential distributions:

$$p_i(x_1, x_2) = c_{i1}c_{i2}\exp(-c_{i1}(x_1 - m_{i1})u(x_1 - m_{i1}) - c_{i2}(x_2 - m_{i2})u(x_2 - m_{i2})), i = 1, 2, \quad (43)$$

where $u()$ is the unit step function. We used $c_{11} = c_{12} = \sqrt{2}/4$, $m_{11} = m_{12} = 0$; and, $c_{21} = c_{22} = 1$, $m_{21} = 4$, $m_{22} = 0$. Figure 4a shows fifty points from each class. Suppose the a priori probabilities of the two classes are equal, then, the decision boundary of the Bayes minimum-error-rate classifier is a triangle, as shown in Figure 10. The error rate of this classifier is 8.36%.

First we examined the AFLC without linguistic rule. The 100 points (shown in Figure 4) were used as training set as well as test set. The leave-one-out [5] method was used, i.e., when a sample was used for testing, it was taken out of the training set. The FCM algorithm was used to find the $\mathbf{z}_k^{(i)}$'s. Using (36) we obtained the initial values $\sigma_1 = 2.3$ and $\sigma_2 = 0.7$, after which the gradient-search algorithm was used to train σ_1, while σ_2 was fixed to be 0.7. All β_k^i's were set equal to unity. Table 1 summarizes the classification results before and after training σ_1, with $M = 1, 2, 4, 8, 16$, and 50 ($M_1 = M_2 = M$). Note that when $M = 50$ the AFLC is the same as the BMECKE. We see that a small M works better than $M = 50$, which justifies the usefulness of the clustering method.

Table 1: Classification results of AFLC:
Number of misclassified samples

	M^*	1	2	4	8	16	50
Before Training	Errors	20	19	22	15	21	21
After Training	Errors	16	8	9	9	8	11

* M is the number of clusters.

Our next experiment is to see how linguistic rules affect the AFLC. We used the following three linguistic rules: (1) *IF* x_1 *is LARGE, THEN* y_1 *is 1 and* y_2 *is 0*, (2) *IF* x_2 *is LARGE, THEN* y_1 *is 1 and* y_2 *is 0*, and (3) *IF* x_1 *is SMALL, THEN* y_1 *is 1 and* y_2 *is 0*. The membership functions for LARGE and SMALL are shown in Figure 9.

We used the Direct Method to generate numerical rules, so that the AFLC without linguistic rules reduces to the BMECKE. In this way, we are able to examine the effects caused solely by linguistic rules. In the experiment, all β's and α's are unity, except for α_3, which was set equal to 3.

Monte Carlo simulations were conducted to test the FL classifier versus the BMECKE. In each run of the simulations, M randomly drawn points for each of the two classes were used in training, and another 250 points from each class were used in testing; 100 runs were conducted for $M=$ 10, 20, 100; and, σ_i were computed by (36) for each run of the simulations. Results are summarized in Table 2, where we also give the results for the case when the same σ was used for the two classes, i.e., σ was computed by (36) when all samples are assumed to come from a single class (in fact, Specht's probability neural network classifier [11], which is essentially a BMECKE, uses the same σ for all classes). It is seen that in all cases, the FL classifier gives better performance than the BMECKE.

Table 2: Classification results of FL and BMECKE classifiers: average percentage of errors of 100 runs

	σ_i different		σ_i same	
M	FLC	BMECKE	FLC	BMECKE
10	11.25	12.81	10.49	16.56
20	12.86	13.77	11.30	14.62
100	9.89	10.38	11.57	13.24

Decision boundaries of the BMECKE and the AFLC for a specific set of training data are shown in Figure 10. The decision boundary of the theoretical Bayes minimum-error classifier is also shown in the figure. Obviously the AFLC approximates the optimal boundary better than the BMECKE. Table 3 summarizes the theoretical error rate of the Bayes classifier and the percentage of misclassified points out of a set of 1000 test samples (500 for each class) for the other two classifiers.

Remark: We observe that linguistic knowledge can have a significant effect on the performance of a FL classifier. A heuristic explanation for this is that linguistic rules function as additional prototypes; therefore, linguistic rules can help to improve performance if the prototype set is small or does not correctly represent the characteristics of the classes. Of course, poor linguistic rules can lower the performance of the classifier.

Table 3: Classification results of three classifiers: Percentage of misclassified test samples

Bayes	FL	Kernel
8.36	14.6	22.6

6 Conclusions

We have developed a method to use fuzzy inference in classifier designs, and have constructed a structure for a FL classifier that utilizes both numerical data and linguistic information. We have shown that, when using certain linguistic rules, the AFLC reduces to the Bayes minimum error classifier; and, when using a certain method to generate fuzzy rules from numerical data, the AFLC reduces to a Bayes classifier that uses Parzen's kernel-estimation of probability densities. We also discussed some methods to determine the parameters in the FL classifier. Our experimental results showed that the FL classifier uses linguistic information in a reasonable way, and that it can provide better performance than probabilistic classifiers that do not use linguistic information.

ACKNOWLEDGEMENT

Portions of this work were performed under a subcontract from XonTech, Inc. to the University of Southern California, and all of this work was supported by the National Science Foundation under NSF Grant MIP-9122018.

References

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

[2] J. C. Bezdek, "Convergence Theory for Fuzzy c-Means: Counterexamples and Repairs," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, no. 5, pp. 873-877, September/October, 1987.

[3] K. Fukunaga, *Introduction to Statistical Pattern Recognition, Second Edition*, Academic Press, New York, 1990.

[4] H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural Network That Learn from Fuzzy If-Then Rules," *IEEE Trans. Fuzzy Systems*, vol. 1, no. 2, pp. 85-97, May 1993.

[5] J. M. Keller and D. J. Hunt, "Incorporating Fuzzy Membership Functions into the Perceptron Algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, no. 6, pp. 693-699, November 1985.

[6] J. M. Keller, M. R. Gray and J. A. Givens, Jr., "A Fuzzy K-Nearest Neighbor Algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 4, pp. 580-585, July/August, 1985.

[7] B. Kosko, *Neural Network And Fuzzy Systems, A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall, Englewood Cliffs, 1992.

[8] R. P. Lippman, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, vol. 4, pp. 4-22.

[9] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Parts 1 and 2," *IEEE Trans. Systems, Man, Cybernetics*, vol. 20, no. 2, pp. 404-435, 1990.

[10] B. W. Silverman, *Density estimation for statistics and data analysis*, London ; New York : Chapman and Hall, 1986.

[11] D. F. Specht, "Probabilistic Neural Networks," *Neural Networks*, vol. 3, pp. 109-118, 1990.

[12] M. Sugeno and T. Yasukawa, "A Fuzzy-Logic-Based Approach to Qualitative Modeling," *IEEE Trans. Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, Feb. 1993.

[13] L. X. Wang and J. M. Mendel, "Back-Propagation Fuzzy Systems as Nonlinear Dynamic System Identifiers", *Proc. IEEE Int. Conf. on Fuzzy Systems*, pp. 1409-1418, 1992.

[14] L. X. Wang and J. M. Mendel, "Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least Squares Learning," *IEEE Trans. Neural Networks*, vol. 3, no.5, pp. 807-814, 1992.

[15] L. A. Zadeh, " Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 1, pp. 28-44, January 1973.

[16] L. A. Zadeh, "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems," *Fuzzy Sets and Systems*, 11, pp. 199-227, 1983.

List of Figures

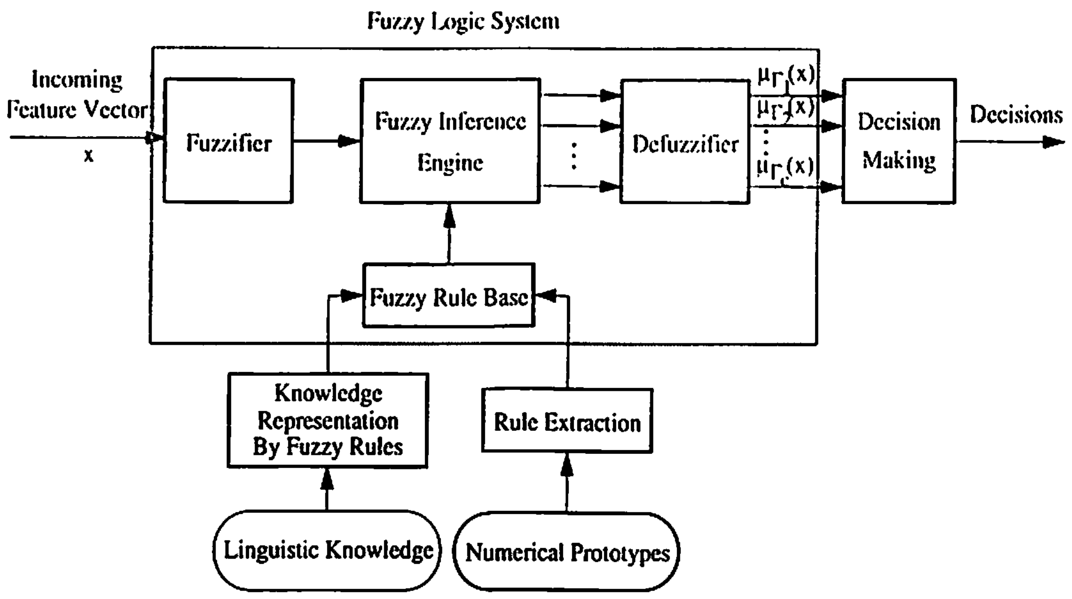

Figure 1: Basic configuration for a FL classifier

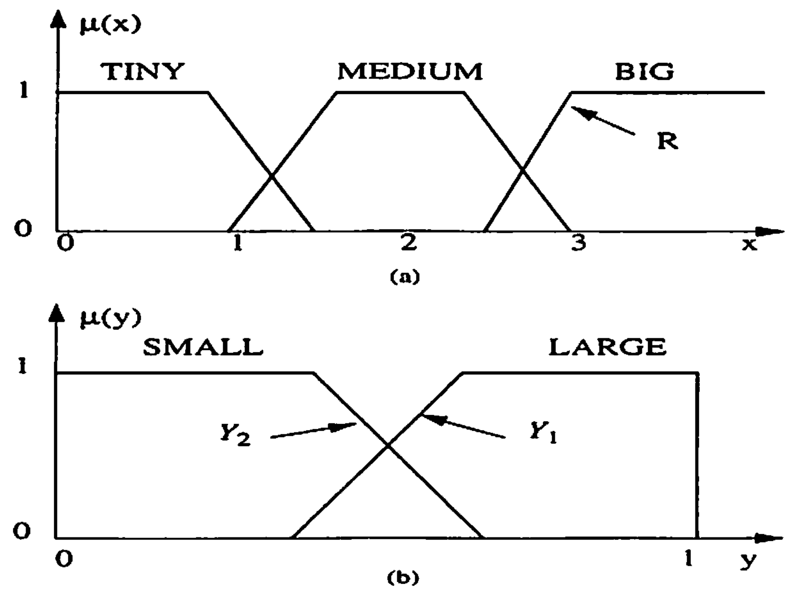

Figure 2: Membership functions of fuzzy sets on the input and output spaces of an FLS. (a) Antecedent fuzzy sets on feature space, and (b) Consequent fuzzy sets on [0,1].

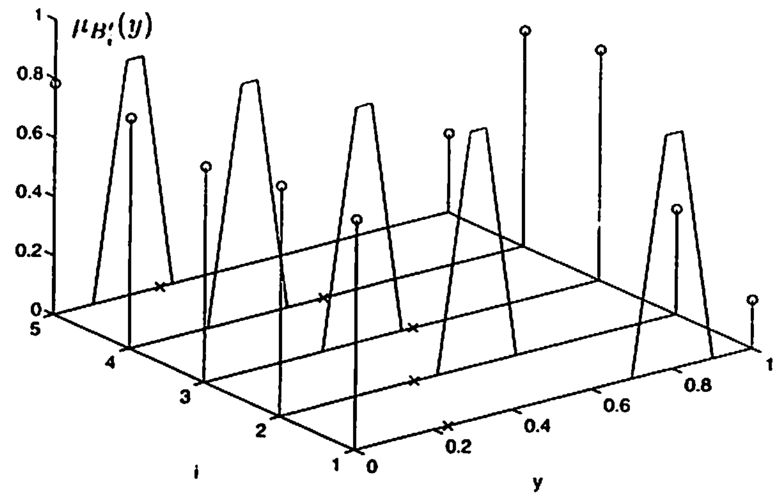

Figure 3: A picture of $\mu_{B_i'}(y)$. Each circle indicates a spike generated by numerical rules, and each trapezoid is an inferred fuzzy set generated by linguistic rules. The y coordinate of the "x" points represents the COA values of B_i'.

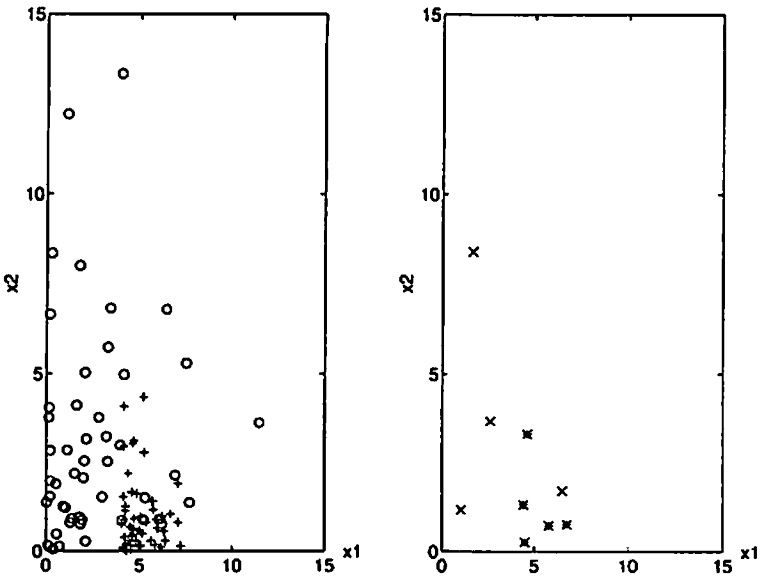

Figure 4: (a) Fifty points from each of two classes, each of which is exponentially distributed; o/+ denotes a point in class 1/class 2; and (b) four FCM centers for each class; x/$\star$ denotes a FCM center of class 1/class 2.

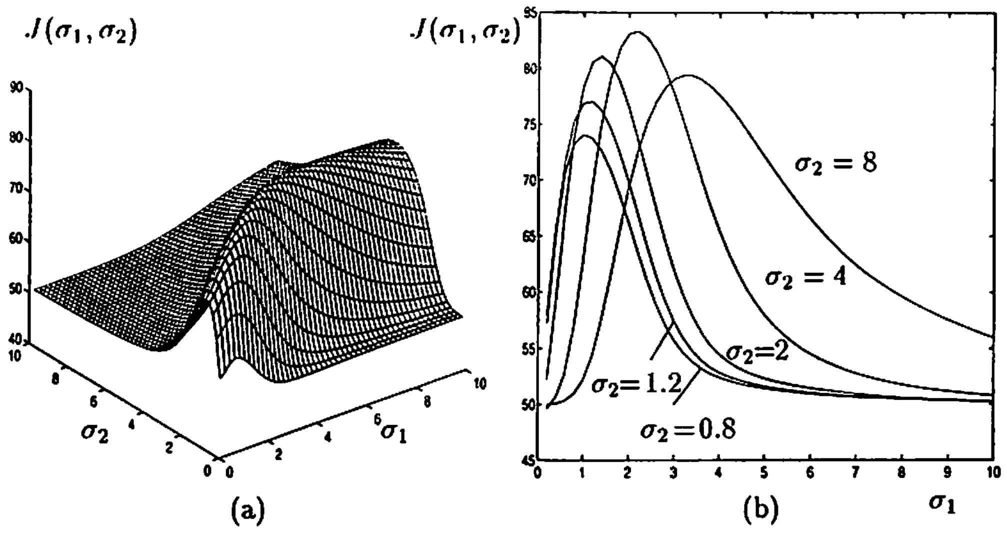

Figure 5: (a) Performance surface of $J(\sigma_1, \sigma_2)$. (b) Performance surface of $J(\sigma_1, \sigma_2)$ for fixed σ_2.

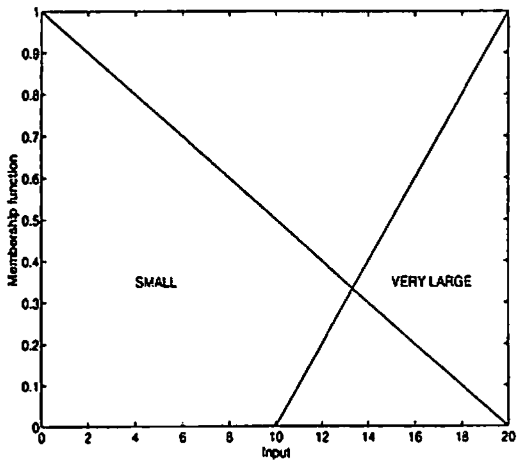

Figure 6: Membership functions for SMALL and VERY LARGE, reproduced from [4]

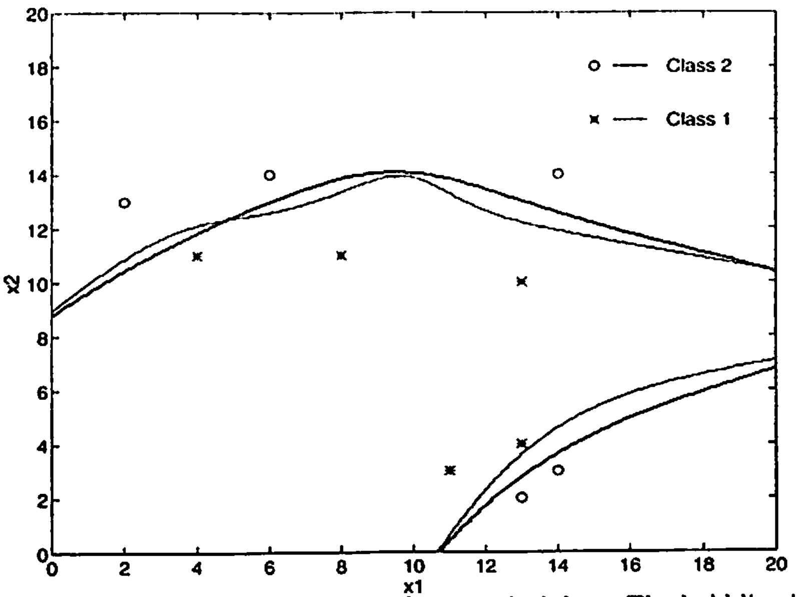

Figure 7: Decision boundaries of BMECKE using only numerical data. The bold line is for $\sigma_i = 5$, and the slim line is for $\sigma_i = 3$.

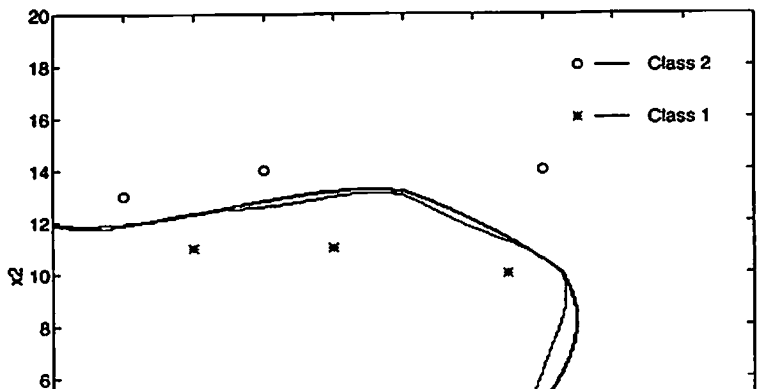

Figure 8: Decision boundaries of AFLC using both numerical data and linguistic rules. The bold line is for $\sigma_i = 5$, and the slim line is for $\sigma_i = 3$.

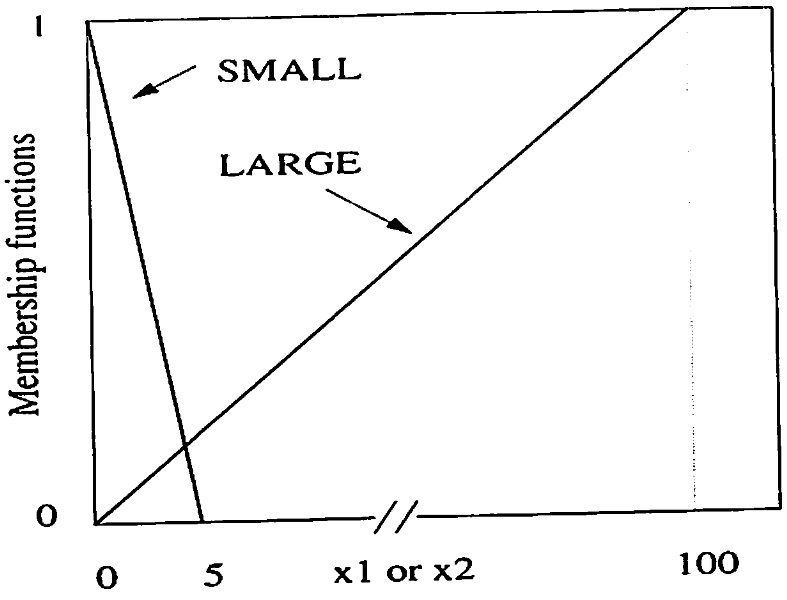

Figure 9: Membership functions of fuzzy sets SMALL and LARGE

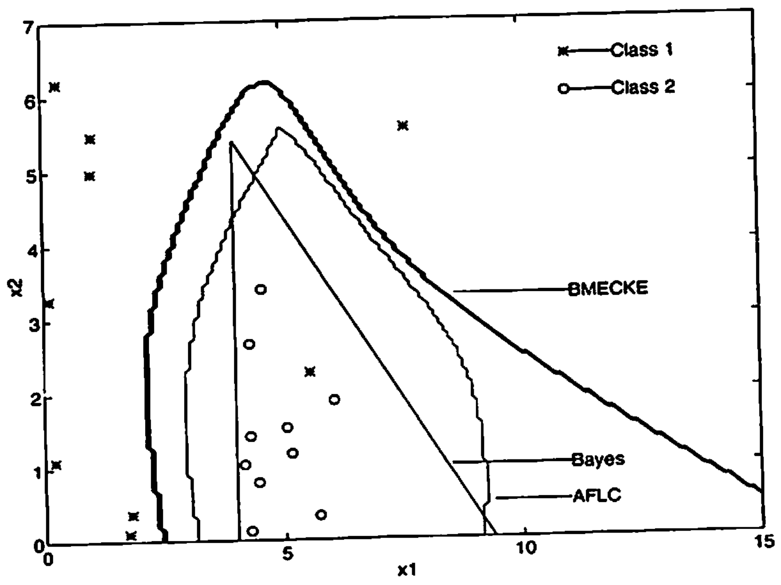

Figure 10: Decision boundaries of the Bayes, BMECKE, and AFLC. BMECKE uses 20 prototypes in kernel estimation; AFLC uses 20 prototypes and three linguistic rules.