Fuzzy Perceptron Neural Networks for Classifiers with Numerical Data and Linguistic Rules as Inputs

Jia-Lin Chen and Jyh-Yeong Chang

***Abstract*—This paper presents a novel learning algorithm of fuzzy perceptron neural networks (FPNNs) for classifiers that utilize expert knowledge represented by fuzzy IF-THEN rules as well as numerical data as inputs. The conventional linear perceptron network is extended to a second-order one, which is much more flexible for defining a discriminant function. In order to handle fuzzy numbers in neural networks, level sets of fuzzy input vectors are incorporated into perceptron neural learning. At different levels of the input fuzzy numbers, updating the weight vector depends on the minimum of the output of the fuzzy perceptron neural network and the corresponding nonfuzzy target output that indicates the correct class of the fuzzy input vector. This minimum is computed efficiently by employing the modified vertex method to lessen the computational load and the training time required. Moreover, the pocket algorithm, called fuzzy pocket algorithm, is introduced into our fuzzy perceptron learning scheme to solve the nonseparable problems. Simulation results demonstrate the effectiveness of the proposed FPNN model.**

***Index Terms*—Fuzzy classifiers, fuzzy functions, perceptron learning.**

I. Introduction

In solving a problem, most scientific algorithms adopt a crisp, or nonfuzzy, discipline and make use of only numerical data because numerical data are easily processed by computers. In this conventional approach, the exclusive processing domain is purely numerical. The number-based approach is usually significant when numerical data are precise enough and representative to the system behavior. This approach usually lacks an ability to model the uncertain or ambiguous information existing among data, which is, however, so often encountered in the real world. On the other hand, humans make many successful decisions and/or judgments primarily on the basis of approximate and/or conceptual information, which is usually uncertain, imprecise, and frequently stated in terms of linguistic terms or rules. Fuzzy set theory has been introduced [1]–[3] to model the uncertain and/or ambiguous characteristics inherent among the data and these characteristics being defined by suitable fuzzy sets and rules are then inferred to reason the useful happening of the result. Since its inception, the research of fuzzy logic has been the focus of various fields and has demonstrated many fruitful results both in theory and application [4]. It can be easily observed that these two paradigms, number-based nonfuzzy approach and fuzzy-logic-based approach, solve problems from different, i.e., in a sense almost complementary, viewpoints. To benefit from these two approaches, a combined routine that integrates both the numerical computation and fuzzy techniques would be more effective than merely uses either one of them. Due to their complementary natures in the way of solving a problem, the integrated scheme will affect the algorithmic routine cooperatively and efficiently and, hence, will enhance the system performance further. As a result, a hybrid paradigm of neuro-fuzzy integration has been a growing area of research in both the academic and industrial communities and has become prevailing in the context of pattern recognition, decision support system, control system applications, and many others [5].

In particular, the realm of designing a classifier system still parallels the above lines of thought. Conventionally, classifier design through numerical data learning is the general approach we have commonly used directly and naturally. For instance, the backpropagation (BP) approach [6], [7] and genetic algorithm [8], [9] are widely used in synthesizing a classifier under the framework of neural models. But these learning activities via datum learning, sequentially count each pattern instance equally without regard to the inherent difference present among the patterns, whereas in another fuzzy-logic-based classification, paradigms emerge recently because they can manipulate the various types of uncertain or ambiguous nature exhibited among the data and can tackle the real-world problems in a manner more like humans. Broadly, the number-based approach proceeds the learning for classification primarily from numerical patterns, but ignores the difference between numerical data, i.e., a collective attribute of the data. Fuzzy set theoretic design conveys the conceptual layout of classifying numerical data given, but considers little on the information in each datum singly. The weakness and lack of collective aspect of the data of a number-based traditional approach is the strength, containing the collective nature of the data set of a fuzzy-logic-based approach. On the contrary, the weakness, lack of the individual nature of each datum of a fuzzy-logic-based approach is the strength, including the character of each training pattern of a number-based approach. To circumvent the defect in the use of the above approaches singly, it is advantageous to integrate these two paradigms together because the weakness of one approach can be counterbalanced by the strength of the other approach and vice versa. Using such a hybrid scheme, number-based and fuzzy-logic-based can enrich the very basic ideas in the framework of classification and can constitute a fundamental ingredient of advanced and successful topology of the classifier.

Manuscript received April 15, 1998; revised October 20, 1999. This work was supported in part by the National Science Council under Grant NSC 85-2213-E-009-119, Taiwan, R.O.C.

The authors are with the Department of Electrical and Control Engineering, National Chiao Tung University, Taiwan 300, R.O.C. (e-mail: jychang@cc.nctu.edu.tw).

Publisher Item Identifier S 1063-6706(00)10692-7.

As was noted before, for connectionist model-based classification systems presented in the literature, most learning procedures utilize and process numerical data only, i.e., each pattern instance is trained sequentially and equally, but the mutual difference existing among them is ignored during the course of training. If, however, other pieces of classification knowledge, especially concerning the nature of patterns in a set, can be included as a part of inputs and then learned by the training procedure, the defect of discarding the pattern difference in learning can be minimized. For instance, fuzzy IF-THEN rules that describe the relation between pattern feature attributes and numerical data in a set or category could be one of many classification domain knowledges that are useful and could be added to describe the system. Linguistic values such as "small," "medium," and "large," are typical and helpful linguistic terms to be defined for specifying patterns in a category by the forms of rules. Under the integrated formalism, the included fuzzy IF-THEN rule inputs will reflect the conceptual layout of the classification problems in a higher level and/or altogether viewpoint. Such conceptual extension definitely enlarges the range of the classification problems and removes the weakness found in the instance training itself. Through integrating fuzzy notions into a traditional number-based classifier, the combined learning scheme will be trained by two kinds of inputs, numerical data of training patterns and structured data of fuzzy classification rules; and they are complementary in nature. Consequently, these two kinds of inputs will affect the learning routine cooperatively and efficiently and the overall classification performance will be enhanced. Such judicious integration matches the increasing trend of deriving a new formulation that can embrace classification schemes involving hybrid numerical and linguistic computation, which is noted in a recent literature review [10].

In the literature to date, two approaches are available for a classifier dealing with linguistic rules and crisp data together. One paradigm extracts fuzzy IF-THEN rules from numerical data and then these deduced rules together with the given linguistic knowledge of rules are combined to execute the classification by fuzzy inference. Corresponding to this paradigm, Wei *et al.* [11] proposed the additive fuzzy logic classifier (AFLC). The AFLC, a direct design scheme without the training phase, does not require a significant amount of learning time needed for a neural-based classifier. However, this approach has some limitations. If only fuzzy IF-THEN rules are used as inputs, the classification results depend on the membership functions of the linguistic labels defined in the if part. Hence, if there exist some regions that are not covered by any linguistic labels of IF-THEN rules, then there is no information to determine an input point in that region to be in which class it belongs to. The other paradigm extends fuzzy notions into the neural network learning for the linguistic rules and then trains all the numerical data and rules by the neural model in a conventional manner [12]–[14]. This approach appears to be more attractive because the neural learning is fused into fuzzy data processing and can learn and generalize from training patterns and fuzzy IF-THEN rules. Following this formalism, Ishibuchi *et al.* [12] proposed a multilayer feedforward neural network to explore the neural learning including fuzzy sets. The learning algorithm is the fuzzy extension of the backpropagation algorithm, referred to hereafter as the FBP algorithm. Based on the extension principle, the learning formula for h-levels of input fuzzy sets are explicitly derived. However, drawbacks of the BP algorithm, such as converging to local minima and/or slow learning convergence, still persist in this BP-based scheme. The shortcoming of being apt to converge to a local minimum causes the FBP algorithm to frequently converge to an inaccurate solution. Also, slow learning convergence leads to a long training time required.

Also in the context of neuro-fuzzy hybrid computing paradigm, a fuzzy neural classifier [15] based on the multilayer perceptron structure and the backpropagation learning algorithm is described. Through converting the numerical/linguistic inputs into larger overlapping linguistic partitions, this model also shows the same feature of capable of handling input vectors presented in quantitative and/or linguistic form, but demonstrates different output forms of providing *outputs of soft belongingness* in terms of degrees of confidence among belonging classes. In this method, the components of the input vector consist of the membership values to the overlapping partitions of linguistic properties, "low," "medium," and "high," corresponding to each input feature. When the input feature is linguistic, its corresponding membership values of the three linguistic terms are quantified as fixed values. The desired output is a membership value denoting the degree of belonging of the input vector to that class. This procedure of assigning fuzzy output membership values, instead of the conventional crisp binary output values, enables this model to be more efficient in classifying ambiguous data with overlapping class boundaries. An extended application of the above scheme is further considered to design a connectionist expert system [16]. In this expert system, the user could be queried for the more essential feature information in case of partial inputs. This expert system also provides justification in the form of rules for any inferred decision.

A most general neuro-fuzzy computing scheme, which is still embedded in a multilayer perceptron structure, was proposed by Hayashi *et al.* [17]. In this fuzzy neural model, both the *input/output signals and weights are all fuzzy sets*. They presented a fuzzified delta rule for learning, however, a method to implement this learning algorithm is still not known. They also argued that a learning algorithm based on h-levels of the error measure is too complicated and may sometimes fail. The difficulty of deriving such general fuzzy functional algorithm through h-levels is obvious.

To provide an efficient and reliable solution, proposed in this article is a new fuzzy neural classification model, which is instead subsumed with crisp outputs and weight parameters (and thus is not as general as the model of Hayashi) and allows inputs either in numerical and/or fuzzy forms. The proposed model is a neural-based learnable classifier, called fuzzy perceptron neural network (FPNN), and its learning scheme is successfully derived based on h-level concept.

The perceptron algorithm [18], [19], a conventional iterative training algorithm, guarantees to determine a linear decision boundary separating the patterns of two classes in a finite number of steps if these patterns are linearly separable. For the linearly nonseparable patterns, Gallant [20], [21] introduced the pocket algorithm to optimally dichotomize the given patterns in

the sense of minimizing the erroneous classification rate. The pocket algorithm is structurally resembled to a conventional perceptron learning except that a checking amendment to stop the algorithm has been added. In light of this concern, this paper incorporates fuzzy sets into a perceptron learning algorithm to enhance the perceptron neural network, which, in addition to handling numerical data, can also handle linguistic knowledge. To avert the limitation of producing a linear boundary by the conventional perceptron, this work introduces a more flexible and simple (under the constraint of limited increase in parameters) boundary by extending the linear discriminant function to a higher order one and, hence, allows a nonlinear separating hyperplane to be generated to tackle nonlinear separability. To this end, we propose a second-order fuzzy perceptron neural network that can handle fuzzy vectors, in a form of fuzzy IF-THEN rules as well as numerical samples as inputs. Based on the level sets of fuzzy numbers, the learning procedure of the fuzzy perceptron network is analyzed. Moreover, the vertex method is modified and applied to find the minimum of the fuzzy discriminant function, whose value indicates whether or not a learning update of the perceptron weight vector should be executed. It is to be noted that in an earlier paper [14], we have proposed a scheme based on level concept and an optimization technique, but it requires much more computational effort in getting the extreme points iteratively. Intensive computational effort needed in the previous paper is greatly reduced by introducing the vertex method in this paper. The pocket algorithm is finally generalized to the fuzzy domain so that the proposed fuzzy perceptron model can copy with a nonseparable case.

It is to be remarked that perceptron learning with a fuzzy membership function can also be found in the literature. Keller and Hunt [22] introduced fuzzy set techniques into the single-layer perceptron algorithm for two-class classification problem. This algorithm assigns fuzzy membership functions to input data to reflect their geometrical proximity to the means of class 1 and class 2 before training the perceptron classifier. This fuzzy perceptron learning scheme can improve the convergence significantly especially when the crisp data are overlapping. The concept and content realized in [22] is quite different from FPNN because it deals with crisp input data only and these data are artificially imposed by membership functions for faster convergence.

The rest of this paper is organized as follows. Section II reviews the concepts of fuzzy function and the extension principle that is employed for analyzing fuzzy functions. Section III introduces the fuzzy perceptron neural networks. Their learning schemes are thoroughly described as well. In Section IV, several numerical examples and the two-spiral benchmark data are simulated. Performance comparisons of the proposed model with other related approaches are summarized by statistical performance evaluation indexes computed from the simulation results. Concluding remarks are finally made in Section V.

II. Fuzzy Function and the Extension Principle

Since our proposed fuzzy perceptron neural network relies heavily on the evaluation of fuzzy function, it is instructive to describe the derivation of the fuzzy function briefly. This section will start with introducing the extension principle, which is the rationale behind evaluating the fuzzy function.

The extension principle [3] is the most important fuzzy set theory that provides the generalization procedure of mapping between fuzzy sets. In light of this principle, algebraic operation on real numbers can be extended to fuzzy numbers, i.e., convex and normal fuzzy sets. According to the extension principle, for a fuzzy multivariable function, $y = f(x_1, x_2, \ldots, x_n)$ of fuzzy variables $A_1, A_2, \ldots, A_n$, i.e.,

$$B = f(A_1, A_2, \ldots, A_n) \tag{1}$$

the membership function of B can be expressed as

$$\mu_B(y) = \operatorname{Sup}_{\substack{x_1 \in A_1, \ldots, x_n \in A_n \\ y = f(x_1, x_2, \ldots, x_n)}} \times \{\min\{\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)\}\}. \tag{2}$$

The computation and algorithm involved in implementing (2) is not trivial to the fuzzy set with a continuous universe. A simple and intuitive way is using the discretization technique [23] in the variable domain. However, if the value of the discretized size is not properly selected, this technique would fail and lead to an irregular and inaccurate result [24], [25]. Consequently, previous investigations [26]–[28] proposed methods for computing fuzzy function, based on the h-level concept. The h-level set is much more effective as a representation form of fuzzy sets since it is a discretization technique on membership value domain of variables, instead of on variable domain themselves. The abnormality of using the conventional discretization on variable domains can be averted by performing the fuzzy function on h-levels. The fuzzy function using h-level concept is illustrated in the following.

For any $h \in [0, 1]$, the h-level sets, i.e., h cuts, of the fuzzy set A are defined as follows:

$$[A]_h = \{x : \mu_A(x) \geq h\}, \qquad \text{for } 0 \leq h \leq 1 \tag{3}$$

where $[\cdot]_h$ denotes an h-level set of a fuzzy set. Furthermore, if $B = f(A_1, A_2, \ldots, A_n)$ is a continuous function and fuzzy sets $A_1, A_2, \ldots, A_n$ are upper semicontinuous, then the following holds[1] [29, p. 39]:

$$[B]_h = f([A_1]_h, [A_2]_h, \ldots, [A_n]_h), \qquad \text{for } 0 \leq h \leq 1. \tag{4}$$

The relation above paves a simpler way to compute the value of a fuzzy function compared with applying the extension principle of (2) directly. In the following, the fuzzy functions encountered in the FPNN learning will be computed by (4) because the above assumptions required by the function and fuzzy sets involved in the FPNN classification tasks are generally satisfied.

III. Fuzzy Perceptron Neural Network

This section first describes the fuzzy IF-THEN rules for classification problems. The proposed neural-based network, i.e., the fuzzy perceptron neural network (FPNN), is then introduced. The FPNN can perform a classification task using not only numerical patterns but also fuzzy IF-THEN rules as

[1]If $[\cdot]_0$ denotes specifically the support of the interval with nonzero membership degrees of a fuzzy set, then (4) holds for $h = 0$ also.

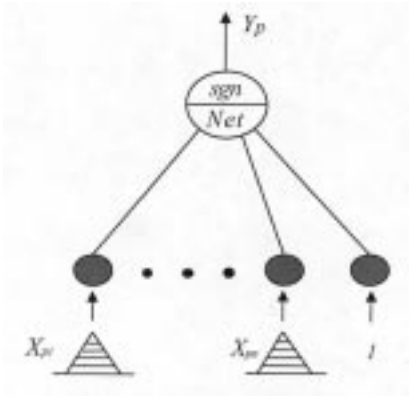

Fig. 1. The architecture of a fuzzy perceptron neural network.

inputs. Next, the vertex method [30] is modified to obtain the minimum of a fuzzy function; this function is the discriminant function of the FPNN. The extremum of the function determines whether or not the coefficients of the discriminant function should be modified in a training iteration. Finally, the fuzzy pocket algorithm is developed to provide a stop criterion for the fuzzy perceptron neural learning.

A. Structure of the Fuzzy Perceptron Neural Network

Based on the perceptron neural network structure, we shall construct a two-class classification system that can also accept fuzzy IF-THEN rules as inputs besides numerical pattern data. Fuzzy IF-THEN rules utilized for the classification problem [12] are given as follows:

$$\begin{aligned}&\text{If } x_{p1} \text{ is } A_{p1} \text{ and } \ldots \text{ and } x_{pn} \text{ is } A_{pm}\\ &\text{then } \mathbf{x}_p = (x_{p1}, \ldots, x_{pn}) \text{ belongs to } C_p\\ &\qquad\qquad p = 1, 2, \ldots, s \quad (5)\end{aligned}$$

where

- A_{pi} linguistic label;
- C_p either class 1 or class 2;
- s number of rules given.

To prevent the discriminant function of the FPNN from passing through the origin only, we can augment the input vectors by including the threshold constant $x_{p(n+1)} = 1$. Because $x_{p(n+1)}$ is a constant of one, it is considered as a fuzzy singleton $A_{p(n+1)} = 1$ to the input vector $\mathbf{x}_p$. With this augmentation, (5) can be generalized and simplified as

$$\begin{aligned}&\mathbf{A}_p = (A_{p1}, \ldots, A_{pn}, A_{p(n+1)}) \text{ belongs to } C_p\\ &\qquad\qquad p = 1, 2, \ldots, s \quad (6)\end{aligned}$$

where $\mathbf{A}_p$ denotes a fuzzy vector.

With the crisp data being regarded as fuzzy numbers of singleton, numerical input data to be classified are considered as a special form of linguistic knowledge represented by (6), in which A_{pi}s are all fuzzy singletons. Therefore, (6) can accommodate the crisp input data set given as well. By this setting, the FPNN is designed, as shown in Fig. 1, so that the augmented $(n+1)$-dimensional fuzzy vectors $\mathbf{A}_p = (A_{p1}, \ldots, A_{pn}, A_{p(n+1)})$ are classified. Each input can be either the linguistic term represented by fuzzy numbers or fuzzy singletons of crisp data. It follows from (4) that the level sets of the output of a fuzzy function can be propagated through the neural network. Fuzzy set A is represented by its level sets: $[A]_{h1}, [A]_{h2}, \ldots, [A]_{hm}$, where $0 \le hj \le 1$, $1 \le j \le m$ and m represents the number of level sets sampled. At these levels, input–output relations of the neural network are derived as follows. At the hj-level of the fuzzy input vector $\mathbf{A}_p, [\mathbf{A}_p]_{hj} = ([A_{p1}]_{hj}, [A_{p2}]_{hj}, \ldots, [A_{pm}]_{hj}, [A_{p(n+1)}]_{hj})$, let level set $\mathbf{X}_p^{hj}$ denotes the interval input vector of $[\mathbf{A}_p]_{hj}$. Then, the fuzzy perceptron neural network for the pth fuzzy IF-THEN rule is defined by

$$\begin{aligned}\text{Input units: } X_{pi}^{hj} &= \left[x_{pi}^{hj\,L}, x_{pi}^{hj\,U}\right]\\ &= [A_{pi}]_{hj}\\ &\qquad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, m. \quad (7)\\ X_{p(n+1)}^{hj} &= [A_{p(n+1)}]_{hj} = 1 \quad (8)\\ \text{Output unit: } Y_p^{hj} &= \operatorname{sgn}\left(\operatorname{Net}_p^{hj}\right)\\ &= \operatorname{sgn}\left(\left[\operatorname{net}_p^{hj\,L}, \operatorname{net}_p^{hj\,U}\right]\right)\\ &= \left[\operatorname{sgn}\left(\operatorname{net}_p^{hj\,L}\right), \operatorname{sgn}\left(\operatorname{net}_p^{hj\,U}\right)\right] \quad (9)\end{aligned}$$

where $\operatorname{sgn}(\cdot)$ and $\operatorname{Net}_p^{hj}$, respectively, represent the signum activation function [5, p. 208] and the input of output neuron Y_p^{hj} as defined in the following.

Owing to the quest for a nonlinear discriminant boundary rather than just a linear one, this work uses a *second-order* perceptron neural network [31], [32]. Subject to a negligible increase in the number of parameters utilized for perceptron learning, a second-order discriminant function can produce various quadratic curves, paraboloids, ellipsoids, and hyperboloids, by varying the coefficients to meet the needed curvature. This flexibility can hopefully accommodate most of the nonlinear boundary needed to discriminate the hybrid data sets. Without loss of generality, the second-order FPNN is illustrated by a two-dimensional (2-D) input vector, which is augmented to $\mathbf{A}_p = (A_{p1}, A_{p2}, A_{p3})$ in which A_{p3} denotes the fuzzy singleton equal to one. At the h-level of the given 2-D input vector $\mathbf{A}_p$, let $\mathbf{X}_p^h$ represents the interval vector of $[\mathbf{A}_p]_h$, i.e., $\mathbf{X}_p^h = (X_{p1}^h, X_{p2}^h, 1) = ([x_{p1}^{h\,L}, x_{p1}^{h\,U}], [x_{p2}^{h\,L}, x_{p2}^{h\,U}], 1)]$. The weighted sum Net_p^h of a second-order perceptron neuron for the input vector $\mathbf{X}_p^h$ is given by

$$\begin{aligned}\operatorname{Net}_p^h &= \left[\operatorname{net}_p^{h\,L}, \operatorname{net}_p^{h\,U}\right]\\ &= w_1\left(X_{p1}^h\right)^2 + w_2 X_{p1}^h X_{p2}^h + w_3\left(X_{p2}^h\right)^2\\ &\quad + w_4 X_{p1}^h + w_5 X_{p2}^h + w_6. \quad (10)\end{aligned}$$

For the interval input vectors $\mathbf{X}_p^h$ with the corresponding target output d_p which is either 1 of class 1 or -1 of class 2, the classifier is required to find the perceptron weight vector $\mathbf{W} = [w_1, w_2, w_3, w_4, w_5, w_6]$ so that

$$\begin{aligned}F_p^h = \left[f_p^{h\,L}, f_p^{h\,U}\right] &= \operatorname{Net}_p^h \cdot d_p > 0\\ &\text{for each } h\text{-level of all the } s \text{ rules.} \quad (11)\end{aligned}$$

As mentioned in Section II, the fuzzy function calculation by the use of the extension principle is equivalent to the h-level-based evaluation by (4) that involves interval arithmetic. Some fundamental properties regarding interval arithmetic [30] are therefore summarized as follows.

Assuming that X_1, X_2, and X_3 are interval numbers, we have the following.

Associativity:

$$X_1 + (X_2 + X_3) = (X_1 + X_2) + X_3$$
$$X_1 \cdot (X_2 \cdot X_3) = (X_1 \cdot X_2) \cdot X_3. \tag{12}$$

Commutativity:

$$X_1 + X_2 = X_2 + X_1$$
$$X_1 \cdot X_2 = X_2 \cdot X_1. \tag{13}$$

However, distributivity does not always hold. Instead, we have the following.

Subdistributivity:

$$X_1 \cdot (X_2 + X_3) \subset X_1 \cdot X_2 + X_1 \cdot X_3. \tag{14}$$

Above equation hints that after the distributing operation, the resultant range of interval would be enlarged. Distributivity fails because two occurrences of an identical interval number X_1 in the right-hand side of (14) are treated as two *independent* interval numbers. With this subdistributivity property in mind, we can see that deriving the interval Net_p^h of (10) is complex since the input variables, X_{p1}^h and X_{p2}^h of the second-order fuzzy perceptron neural network appear more than once and should be treated as linearly *dependent*. Hence, to evaluate Net_p^h of (10) in a manner as direct interval arithmetic computation in the right hand side of (14) is inappropriate.

To satisfy inequality (11) at each h-level of the fuzzy input vector, we have to invent a scheme that can search for the extremum of the Net_p^h function for a given weight vector $\mathbf{W}$. If the target output is class 1, then we set $d_p = 1$ and we will find the minimum $\mathrm{net}_p^{h^L}$. On the contrary, we set $d_p = -1$ and we will find the maximum $\mathrm{net}_p^{h^U}$. In fact, only the constrained minimization must be found because for the case $d_p = -1$, the maximization is transformed to the minimization by negating the Net_p function. At the h-level of fuzzy sets involved in the pth IF-THEN rule, the problem becomes to find the minimum of $\mathrm{Net} = w_1X_1^2 + w_2X_1X_2 + w_3X_2^2 + w_4X_1 + w_5X_2 + w_6$ with the constraints $X_1 = [x_1^L, x_1^U]$ and $X_2 = [x_2^L, x_2^U]$, which define the subspace of the 2-D parameters and are called the feasible region of X_1 and X_2. Notably, Net X_1 and X_2 are p and h dependent, i.e., $\mathrm{Net} = \mathrm{Net}_p^h$, $X_1 = X_{p1}^h$, and $X_2 = X_{p2}^h$. However, for simplicity, these two scripts are not explicitly shown in Net X_1 and X_2. Although our previous work [14] derived an optimization scheme to solve the above problem, it is, however, computationally expensive. In the sequel, a new computationally efficient technique, named the modified vertex method is introduced to solve this constrained minimization problem.

B. Fuzzy Perceptron Learning by the Modified Vertex Method

As mentioned earlier, solving a constrained minimization problem is necessary for the fuzzy perceptron neural learning. According to our observation, this minimization problem can be solved more efficiently by modifying and applying the vertex method. Details of this approach are as follows.

1) Vertex Method: Let f be an n-dimensional interval function given by

$$\begin{aligned} Y &= f(X_1, X_2, \ldots, X_n) \\ &= \{f(x_1, x_2, \ldots, x_n) \,|\, x_1 \in X_1 \\ &\quad x_2 \in X_2, \ldots, x_n \in X_n\} \end{aligned} \tag{15}$$

where

$$X_i = \left[x_i{}^L, x_i{}^U\right], \qquad i = 1, \ldots, n. \tag{16}$$

Function value Y is also an interval number. These n interval variables form an n-dimensional hypercube $X_1 \times X_2 \times \cdots \times X_n$ with N, i.e., 2^n, vertices. All vertices' coordinates of the interval function are a combination of n pairs of end points of interval numbers. According to the vertex method [30], these vertices in the n-dimensional space are critical to calculate the interval of a function of interval variables. Essential properties of the vertex method are given as follows.

For a continuous and differentiable function f in the n-dimensional hypercube, if f has no extreme point, i.e., a point with a differential value equal to zero, in the feasible region, interval of the function in the defined domain (including the boundaries) can be obtained by

$$\begin{aligned} Y &= f(X_1, X_2, \ldots, X_n) \\ &= [\min\{f(v_i) \,|\, i \in \{1, \ldots, N\}\} \\ &\quad \max\{f(v_j) \,|\, j \in \{1, \ldots, N\}\}] \end{aligned} \tag{17}$$

where v_i and v_j denote the ith and jth vertices of the N vertices of the hypercube.

The vertex method is effective only when the conditions of continuity and no extreme point existing in the region are satisfied. Furthermore, if extreme points of the function f exist in the feasible region, these extreme points must also be checked to obtain the minimal value. That is, suppose that function f has Q extreme points, then interval calculation of (17) is extended to

$$\begin{aligned} Y = [&\min\{f(v_i), f(E_k) \,|\, i \in \{1, \ldots, N\}, k \in \{1, \ldots, Q\}\}, \\ &\max\{f(v_j), f(E_l) \,|\, j \in \{1, \ldots, N\}, l \in \{1, \ldots, Q\}\}] \end{aligned} \tag{18}$$

where E_k and E_l denote the kth and lth extreme points.

In light of results above, how to determine the minimum of $\mathrm{Net} = w_1X_1^2 + w_2X_1X_2 + w_3X_2^2 + w_4X_1 + w_5X_2 + w_6$ with the constraints $X_1 = [x_1^L, x_1^U]$ and $X_2 = [x_2^L, x_2^U]$ is considered again. Fig. 2 depicts the case that the extreme point, denoted as $(x_1, x_2)_e$, exists in the feasible region, while Fig. 3 shows the case that the extreme point is outside the feasible region. According to our study, the vertex method should be modified when it is exploited in our FPNN model. It can be observed from Figs. 2 or 3 that the minimal point of a Net function could also be located on the boundary of the fuzzy set in addition to the vertices and extreme point. In response to this modification, an iterative searching process by the bisection method [33, ch. 2]

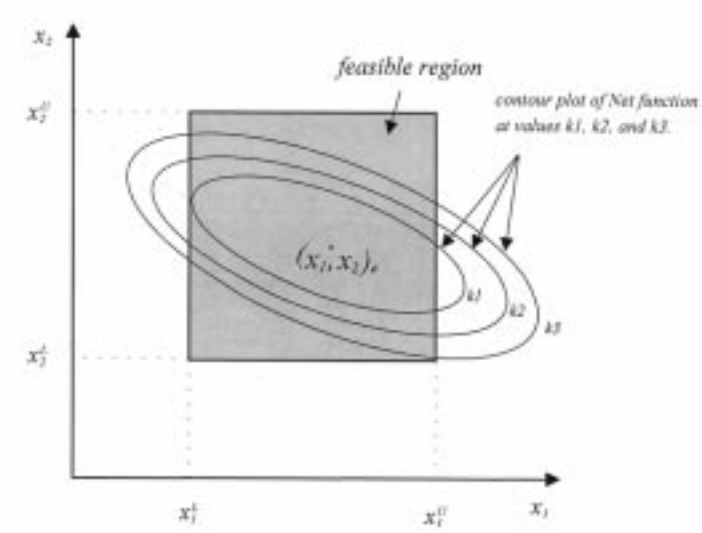

Fig. 2. Searching for the minimum of $\mathrm{Net}(x_1, x_2)$, $x_1^L \le x_1 \le x_1^U$, $x_2^L \le x_2 \le x_2^U$. The extreme point $(x_1, x_2)_e$ is in the feasible region.

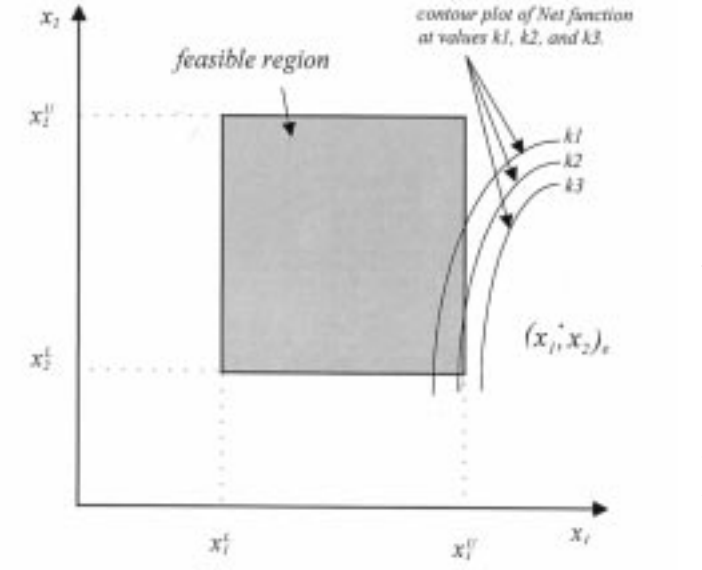

Fig. 3. Searching for the minimum of $\mathrm{Net}(x_1, x_2)$, $x_1^L \le x_1 \le x_1^U$, $x_2^L \le x_2 \le x_2^U$. The extreme point $(x_1, x_2)_e$ is not in the feasible region.

is introduced. For the region defined by the X_1 and X_2 intervals, four boundaries should be checked for the minimal point. For any one of the four boundaries, function f is reduced to an one-dimensional function $f(X_i), i$ is either 1 or 2, with the other variable remaining fixed at its corresponding lower or higher value. For convenience, allow the interval of the changing variable to be defined in $[a, b]$, i.e., $[a, b]$ can be either $[x_1^L, x_1^U]$ or $[x_2^L, x_2^U]$.

2) Bisection Method: To search for the minimal value of the quadratic function $f(X)$ above, with the given interval $[a, b]$ and the maximum number of iterations I, we proceed the following steps.

Step 1) Set $i = 1, \gamma = a, \delta = b$.
Step 2) While $i \le I$, perform Steps 3)–5).
Step 3) Set $\rho = (\gamma + \delta)/2$.
Step 4) $i = i + 1$.
Step 5) If $f(\gamma) \le f(\delta)$, then $\delta = \rho$, else $\gamma = \rho$.
Step 6) The minimum of $f(X)$ is $\min(f(\gamma), f(\delta), f(\rho))$.

The above bisection process is operated on these four boundaries sequentially to get their respective minima and then the smallest value f^* is selected from these four minima obtained. Finally, the minimal Net value $\mathrm{Net}_p^{h^*}$ equals f^* if there is no extreme point in the feasible region, otherwise $\mathrm{Net}_p^{h^*}$ is the minimum of f^* and the Net values evaluated at the extreme points in the feasible region. In addition, the corresponding minimal point is denoted as $(x_{p1}^{h\,*}, x_{p2}^{h\,*})$. Parameter I determines how accurate the minimal point is. For instance, in this paper, six iterations are selected to locate the optimal point $(x_{p1}^{h\,*}, x_{p2}^{h\,*})$ leading to an accuracy of $(b-a)/2^6$. For the h-level of the pth fuzzy input vector, if $\mathrm{Net}_p^{h^*} \cdot d_p < 0$, then the second-order fuzzy perceptron learning algorithm with a learning constant μ updates the weight vector $\mathbf{W}$ by

$$\mathbf{W}' = \mathbf{W} + \mu \cdot h \cdot d_p \cdot \mathbf{z}_p^{h^*} \tag{19}$$

where

$$\mathbf{z}_p^{h^*} = \left[x_{p1}^{h\,*2}, x_{p1}^{h\,*} \cdot x_{p2}^{h\,*}, x_{p2}^{h\,*2}, x_{p1}^{h\,*}, x_{p2}^{h\,*}, 1\right]. \tag{20}$$

The above learning procedure does not stop until inequality (11) holds for all h-levels of s rules and all the crisp training patterns as well. The learning step size of the above fuzzy perceptron algorithm is proportional to the current h-level. A larger membership degree implies a larger learning step size. This learning procedure can accept not only fuzzy IF-THEN rules, but also crisp data since real numbers can be regarded as fuzzy singletons and the corresponding h-level is assumed to equal one. If the training data are crisp, the proposed second-order fuzzy perceptron network reduces to the conventional second-order perceptron network. Furthermore, the proposed scheme is quite general since it can be easily extended to the third- or higher order fuzzy perceptron model in the same manner. From this perspective, the proposed second-order fuzzy perceptron algorithm can be viewed as an extension of the conventional perceptron learning to the case of fuzzy rules as inputs.

C. Fuzzy Pocket Algorithm

Perceptron learning is quite appropriate for separable problems, i.e., problems for which some set of weights is available that correctly classifies all training patterns after a finite number of mistakes. Nonseparable problems are a different story. The fact that no set of weights can correctly classify all training patterns implies that a set of weights, which correctly classifies as large a fraction of the training patterns as possible is preferred. Pocket algorithm is developed to determine a set of weight in this optimal sense [20]. In line with such optimal sense, a fuzzy pocket algorithm is developed to resolve the nonseparable problem encountered in fuzzy perceptron learning.

This work modifies the pocket algorithm to effectively address the nonseparable case for our fuzzy perceptron algorithm such as overlapping fuzzy numbers or the input data cannot be dichotomized by a second-order discriminant function. The pocket algorithm adds additional steps to monitor the

performance of the perceptron network. For training patterns of crisp data, the performance is measured on the basis of the correct classification among training patterns. The pocket algorithm must be modified so that it can be applied involving fuzzy input vectors. This modified pocket algorithm called fuzzy pocket algorithm should optimally dichotomize the fuzzy input vectors of linguistic terms as follows.

1) The total misclassified membership values of the two classes should be as small as possible.
2) The difference between the maximal misclassified membership values (the highest misclassified membership value of overlapping fuzzy input vectors for each class) of these two classes should be as small as possible.

At the h-level of fuzzy numbers, the following index, *mem_index*, should be minimized to find an optimal weight vector that can realize the above two statements

$$mem\_index = mem\_mis + \beta \cdot (1 - \eta^{-k}) \cdot |mem\_dif| \tag{21}$$

where *mem_mis* denotes the sum of the membership levels of those training vectors which cannot be correctly classified by the discriminant function and $|mem\_dif|$ represents the absolute value of the difference between the maximal misclassified membership levels of the two classes. To be fair to both misclassified classes occurring in overlapping fuzzy input vectors, the decision boundary should be located at a point that the maximal misclassification membership functions of the misclassified classes are as nearly equal as possible and this justifies the requirement of a minimal $|mem\_dif|$ value. Parameter k is the index of iteration number in the training procedure, whose value goes from one up to the epoch number we selected. Constant η, a value greater than one but very close to one, increasingly emphasizes *mem_dif* in accordance with an increasing number of iterations. Constant β, which can certainly be chosen by the user, is introduced for the relative weighting of *mem_mis* and $|mem\_dif|$. By our experience, one, two, or three are suitable choices for constant β. In the numerical simulations presented later, we used $\beta = 2$ for all the illustrative examples. In this manner, the fuzzy pocket algorithm identifies a weight vector that can minimize mainly the cumulative misclassified membership values of fuzzy numbers initially and then searches for a weight that cannot only minimize the misclassification error but also minimize the difference between maximal misclassification membership values. Notice that in the case of nonoverlapping fuzzy numbers, *mem_index* is calculated from *mem_mis* only, while $|mem\_dif|$ is neglected. In a manner resembling the pocket algorithm, we save "in the pocket" of the weight vector with the smallest *mem_index* in the fuzzy perceptron algorithm. This simple modification facilitates fuzzy perceptron learning well behaved.

The relative weighting between crisp data and linguistic rules on calculating *mem_mis* is worth mentioning. This relative weighting is subjective and, naturally, more reliable information (either crisp data or linguistic rules) should be more emphasized. In this paper, six h-levels, i.e., $h = 0, 0.2, 0.4, 0.6, 0.8,$ and 1, are used in the fuzzy perceptron learning algorithm, whereas for crisp data, only h-level of one is assigned. On *mem_mis* calculation, the relative weighting for a crisp datum is chosen three times that of a fuzzy IF-THEN rule. This relative weight compensates the crisp data for using only h-level of one instead of the six h-levels of rules. By this setting, we count equally on numerical data and the linguistic rules because the sum of these six levels equals three.

To summarize, the second-order fuzzy perceptron learning with fuzzy pocket algorithm is as follows.

1) Set $\mathbf{W}$ to a small and random vector.
2) Let $\mathbf{W}$ be the current weight. Select a training fuzzy input vector $\mathbf{X_p}$.
3) If $\mathbf{W}$ correctly classifies $\mathbf{X_p}$, then
   a) If the current run of the *mem_index* is smaller than the run of *mem_index* in your pocket, then put $\mathbf{W}$ in your pocket and remember the *mem_index* of its run.

   else form a new set of weights $\mathbf{W}'$ by fuzzy perceptron learning.
4) If the specified number of iterations has not been taken or the specified *mem_index* has not been reached, then go to 2); otherwise, stop the iteration.

D. Multiclass Classification

To be more general than just dealing with two-class classification [14], the proposed FPNN model can be extended to solve the multiclass classification problems by increasing the number of discriminant functions equal to the number of classes to be classified. For a c-class classification problem, we define c discriminant functions $\mathrm{Net}_{p,i}^{h}$ with weights $\mathbf{W}_i$, $i = 1, 2, \ldots, c$. At the h-level of the pth fuzzy input vector $\mathbf{X}_p^h$, if $\mathbf{X}_p^h$ belongs to class i, then $\mathrm{Net}_{p,i}^{h}{}^{*}$ value should be the largest among the c discriminant functions. If, however, for some j, we have $\mathrm{Net}_{p,i}^{h}{}^{*} < \mathrm{Net}_{p,j}^{h}{}^{*}$, then the updating rules for weight vectors are given by

$$\begin{aligned} \mathbf{W}_i' &= \mathbf{W}_i + \mu \cdot h \cdot \mathbf{z}_p^{h*} \\ \mathbf{W}_j' &= \mathbf{W}_j - \mu \cdot h \cdot \mathbf{z}_p^{h*} \\ \mathbf{W}_l' &= \mathbf{W}_l, \qquad \text{for } l = 1, 2, \ldots, c,\ l \neq i, j \end{aligned} \tag{22}$$

where $\mathbf{z}_p^{h*}$ is in the same form of (20).

For a multiclass nonseparable problem, the modified pocket algorithm should be amended to suit the concept mentioned in the above subsection. The *mem_index* should be revised to the following form:

$$mem\_index = \sum_{i=1}^{c} mem\_mis(i) + \beta \cdot (1 - \eta^{-k}) \cdot \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} |mem\_dif(i,j)| \tag{23}$$

where *mem_mis*(i) is the sum of erroneous membership levels of the ith class patterns, i.e., $\{h \,|\, \mathrm{Net}_{p,i}^{h}{}^{*} < \mathrm{Net}_{p,j}^{h}{}^{*}$, for all h-levels of those pth rules defined for the ith class but erroneously classified to the jth class$\}$. The $|mem\_dif(i,j)|$ in (23) is the absolute value of the difference between the maximal misclassification membership levels of classes i and j in the i, j class overlapping region. Via this modification, the fuzzy pocket

algorithm can be extended to multiclass classification problems in a manner analogous to two-class problems.

IV. Simulation

Simulations were performed not only to verify the effectiveness of the proposed fuzzy perceptron neural network, but also to compare with that obtained by FBP and AFLC algorithms. In this section, two simulations were presented. In Simulation 1, all the fuzzy IF-THEN rules for classification problems were initially given. Four testing examples were provided; the first three examples were the two-class classification problems and the final one addressed three-class classification. Notice that the results of Examples 2 and 4 by AFLC were not presented because these two examples are not solvable by AFLC since undefined regions of fuzzy inputs exist. In Simulation 2, we tested the proposed algorithm on the neural network benchmark problem, two-spiral classification proposed by Lang *et al.* [34]. In this simulation, we generated IF-THEN rules from the two-spiral data to underline the existing regularity among samples. The FPNN classification boundaries were improved by incorporating these rules generated and crisp data as inputs.

A. Simulation 1

Due to the nondeterministic learning nature of FBP and FPNN each of these four examples was run for 200 design trials. Based on these running results, classification performance in terms of several statistical indexes will be provided in the subsequent subsection. For these two algorithms, a learning cycle of each pattern and every fuzzy IF-THEN rule being presented once constitute an *epoch* of learning iteration.

In running the four examples by the FBP algorithm, the feedforward neural network was structured with one hidden layer of five hidden units; these network structures and the number of epochs were chosen as recommended (except Example 3) by [12]. In our experience, the FBP algorithm cannot easily converge to satisfactory solutions. As a consequence, there are few satisfactory decision boundaries obtained in the 200 trials of each example. Therefore, in the following illustrative examples, the best decision boundary (in the sense of minimal error of FBP [12]) of each example was selected from 200 design trials and then plotted.

For FPNN, the following examples were simulated by employing six levels of h, i.e., $h = 0, 0.2, 0.4, 0.6, 0.8$, and 1.0 for the linguistic values in the fuzzy perceptron learning algorithm. Note that to make the zero level of the fuzzy number effective hinges on replacing $h = 0$ with $h = 0.05$ (a small positive value) whenever updating $\mathbf{W}$ from (19) or (22), as well as computing the *mem_index* of (21) or (23). In running FPNN with 200 design trials for each example, almost all decision boundaries obtained are similar to the best one (still sticking to the sense of minimal *mem_index*). Also, we plotted the best decision boundary of FPNN algorithm for each of these four examples.

Example 1: In line with Ishibuchi *et al.* [12], this work designed a two-class classifier on a pattern space $[0, 20] \times [0, 20]$. The numerical data are

$$\text{class } 1 = \{(4, 11), (8, 11), (11, 3), (13, 4), (13, 10)\} \quad (24)$$

$$\text{class } 2 = \{(2, 13), (6, 14), (13, 2), (14, 3), (14, 14)\}. \quad (25)$$

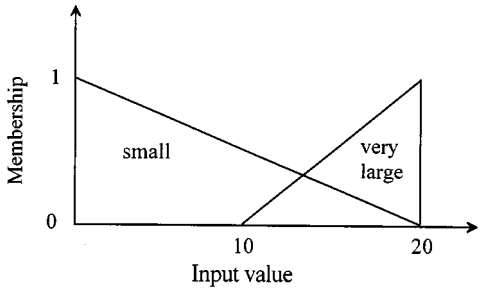

Fig. 4. The membership functions of the linguistic values "small" and "very large" in Example 1.

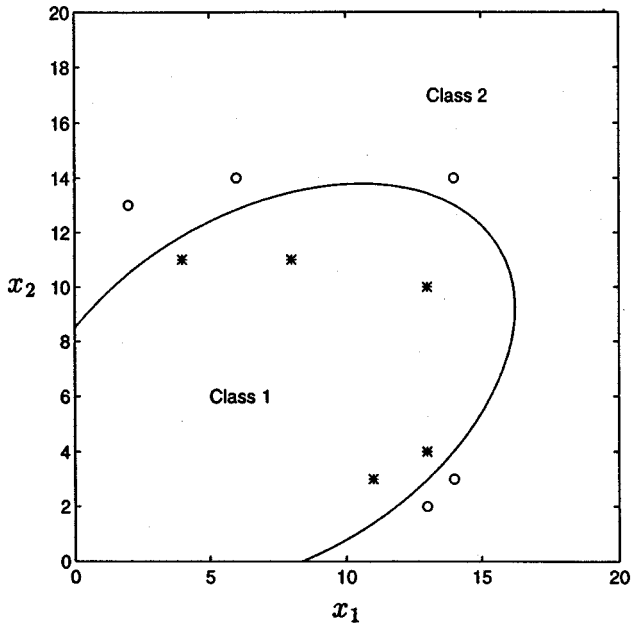

Fig. 5. A simulation result of learning with only numerical data by the proposed second-order fuzzy perceptron learning algorithm.

The following two fuzzy IF-THEN rules given from human experts are

$$\begin{aligned}&\text{If } x_{p1} \text{ is small and } x_{p2} \text{ is small}\\ &\text{then } \mathbf{x}_p \text{ belongs to class 1.}\end{aligned} \quad (26)$$

$$\begin{aligned}&\text{If } x_{p1} \text{ is very large or } x_{p2} \text{ is very large}\\ &\text{then } \mathbf{x}_p \text{ belongs to class 2.}\end{aligned} \quad (27)$$

Fig. 4 displays the membership functions, which are adopted from [12] and coincide with our intuition, of the fuzzy numbers "small" and "very large." The fact that the pattern space is $[0, 20] \times [0, 20]$ accounts for why the fuzzy IF-THEN rule (27) with the "or" connection can be converted into the following two rules with the "and" connection:

$$\begin{aligned}&\text{If } x_{p1} \text{ is very large and } x_{p2} \text{ is in } [0, 20]\\ &\text{then } \mathbf{x}_p \text{ belongs to class 2.}\end{aligned} \quad (28)$$

$$\begin{aligned}&\text{If } x_{p1} \text{ is in } [0, 20] \text{ and } x_{p2} \text{ is very large}\\ &\text{then } \mathbf{x}_p \text{ belongs to class 2.}\end{aligned} \quad (29)$$

Initially, we trained the second-order fuzzy perceptron using *only numerical data*. Fig. 5 plots the simulation result with $\mu =$

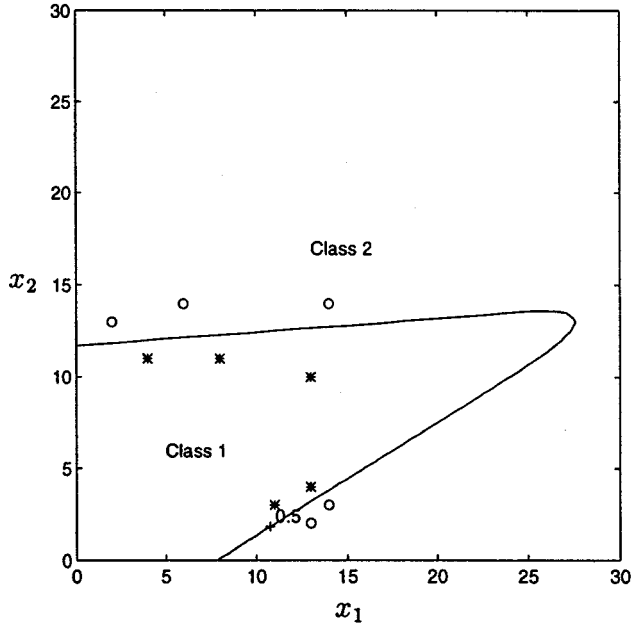

Fig. 6. The simulation result by the FBP algorithm with only numerical data.

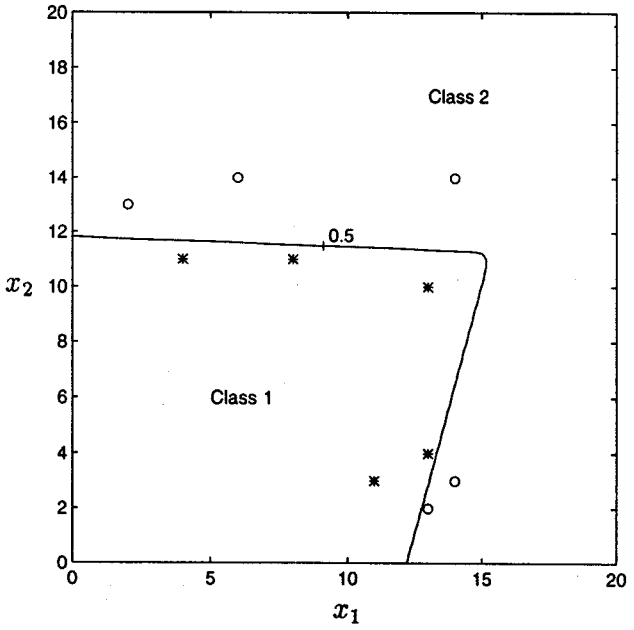

Fig. 8. The simulation result of Example 1 using the FBP algorithm.

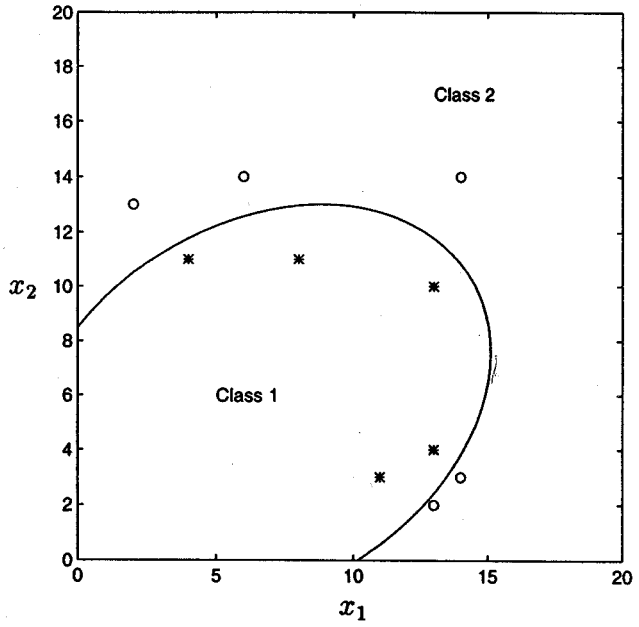

Fig. 7. The simulation result of learning with both numerical data and fuzzy IF-THEN rules by the proposed method.

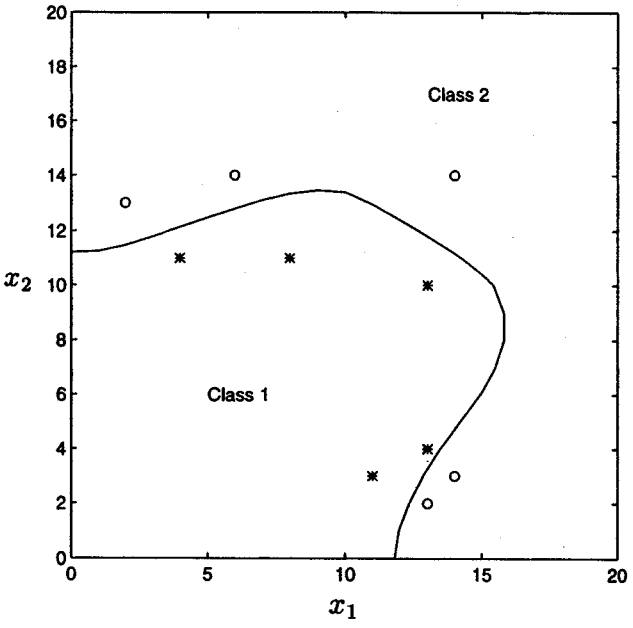

Fig. 9. A simulation result of Example 1 using the AFLC approach.

0.1 after 100 epochs training. According to this figure, all the given patterns are correctly classified by the second-order fuzzy perceptron neural network. By using only the numerical data and iterating for 1000 epochs, Fig. 6 displays the result obtained from the FBP algorithm. The boundary curve in Fig. 6 is drawn by plotting all the points in the pattern space where the output values are 0.5. This algorithm classifies a test pattern with very large values of x_1 as class 1 [12]. The proposed scheme resolved this weakness because the parameters of a second-order perceptron network are so flexible that the neural network can correlate well with the crisp data.

Next, we trained the second-order fuzzy perceptron neural network using not only numerical data but also fuzzy IF-THEN rules. Fig. 7 depicts the best simulation result with $\mu = 0.1$, $\beta = 2$, and $\eta = 1.01$ for 100 epochs. As this figure reveals, the discriminant boundary classifies all the given patterns and precisely conveys the effects of the two linguistic rules. Among the 200 simulation results by the FBP approach, it is rare to obtain a satisfactory decision boundary. Fig. 8 presents the best decision boundary chosen from 200 FBP trials after 1000 epochs. For the AFLC approach, Fig. 9 summarizes the result of setting the parameters $\sigma = 3$ and all αs and βs equal to unity (where σ, αs,

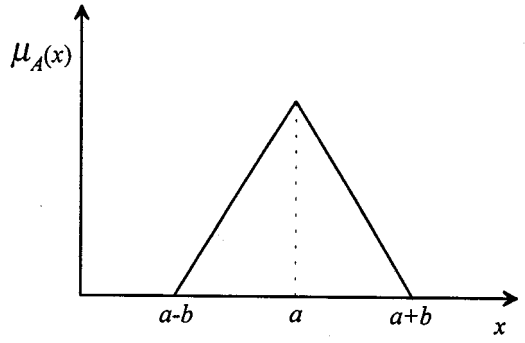

Fig. 10. A symmetric triangular fuzzy number.

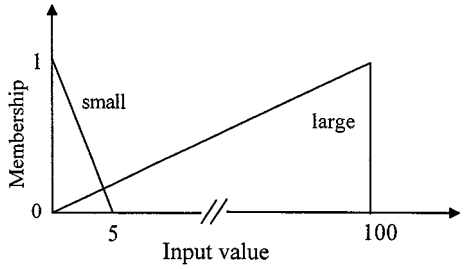

Fig. 11. The simulation result of learning for fuzzy input data vectors by the proposed method. The rectangles are supports of fuzzy vectors.

and βs are the chosen parameters of AFLC) [11]. However, the decision boundary obtained by other parameter settings is not as successful as that shown in Fig. 9.

Example 2: The classification power of the proposed method was examined as a nonlinear classification machine of fuzzy input vectors. The proposed method was applied to the fuzzy data [12]

$$\begin{aligned}\text{class 1} = \{&((3,2)_L,(3,2)_L),((3,2)_L,(11,3)_L),\\ &((9,3)_L,(5,2)_L),((10,2)_L,(10,2)_L)\}\end{aligned} \tag{30}$$

$$\begin{aligned}\text{class 2} = \{&((6,3)_L,(18,2)_L),((12,2)_L,(12,2)_L),\\ &((14,3)_L,(17,2)_L),((16,2)_L,(5,4)_L)\}\end{aligned} \tag{31}$$

where $(a,b)_L$ denotes a symmetric triangular fuzzy number, as shown in Fig. 10, with the center a and the spread b defined by the membership function

$$\mu_A(x) = \max\left\{1 - \frac{|x-a|}{b}, 0\right\}. \tag{32}$$

After 200 epochs and with $\mu = 0.1$, $\beta = 2$, $\eta = 1.01$, Fig. 11 summarizes the best simulation result of a second-order fuzzy perceptron neural network in which the nonlinear boundary curve is highly successful. In this figure, the hatched area and white area denote the supports of these two classes of fuzzy vectors, respectively. This same figure reveals that the given nonoverlapping fuzzy vectors are all correctly classified. In addition, the boundary curve crossing the overlapping area of the given fuzzy vectors was rather fair for both classes. After 1000 epochs of training, Fig. 12 shows the optimally classified simulation result of this example selected from the FBP approaches. Unfortunately, this satisfactory separating boundary is seldom observed during 200 design trials.

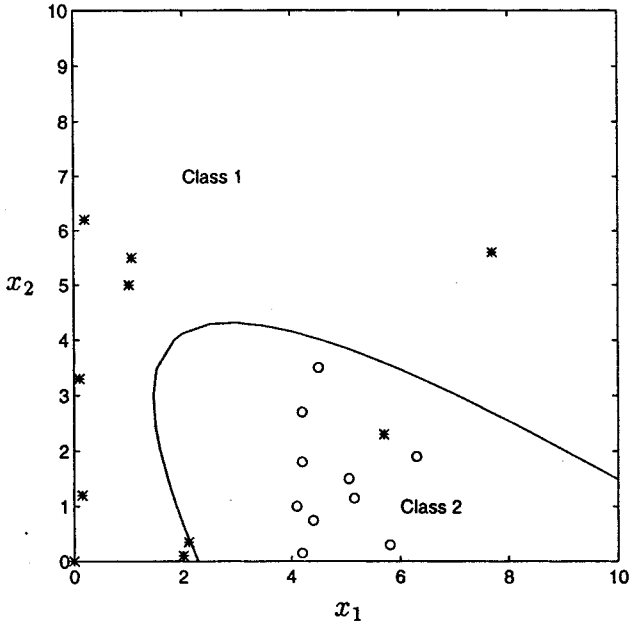

Fig. 12. The simulation result of Example 2 using the FBP approach.

Example 3: Based on an example from [11], we designed a two-class classifier. The numerical training data consist of ten points from each of the two classes

$$\begin{aligned}\text{class 1} = \{&(0.1,3.3),(0.15,1.2),\\ &(0.2,6.2),(1,5),(1.05,5.5),(2,0.1),\\ &(2.1,0.35),(5.7,2.3),(7.7,5.6),(0.0,0.0)\}\end{aligned} \tag{33}$$

$$\begin{aligned}\text{class 2} = \{&(4.2,0.15),(4.4,0.75),\\ &(4.1,1),(4.2,1.8),(4.2,2.7),(4.5,3.5),(5.05,1.5),\\ &(5.15,1.15),(5.8,0.3),(6.3,1.9)\}.\end{aligned} \tag{34}$$

In addition, the following three linguistic rules are used:

$$\text{If } x_{p1} \text{ is large then } \mathbf{x}_p \text{ belongs to class 1} \tag{35}$$

$$\text{If } x_{p2} \text{ is large then } \mathbf{x}_p \text{ belongs to class 1} \tag{36}$$

$$\text{If } x_{p1} \text{ is small then } \mathbf{x}_p \text{ belongs to class 1.} \tag{37}$$

The membership functions for "large" and "small," adopted from [11], are displayed in Fig. 13.

After 2500 epochs training of this example, FPNN with $\mu = 0.1$, $\beta = 2$, and $\eta = 1.01$ leads to the best result shown in Fig. 14. Fig. 15 depicts the least error solution chosen from iterating FBP for 12 000 epochs. Obviously, our approach yields a better boundary than that from the FBP. Good parameters setting of $\sigma = 3.66$ and all αs and βs equal to unity except $\alpha_3 = 3$ for the AFLC scheme [11] produced the decision boundary of Fig. 16. Note that outlier training pattern, $(5.7, 2.3)$, is so atypical that all the three methods cannot classify it correctly.

Example 4: In the following, we considered the three-class classification problem [12] of fuzzy vectors:

$$\begin{aligned}\text{class 1} = \{&((4,1)_L,(17,2)_L),((6,2)_L,(11,2)_L)\\ &((10,3)_L,(16,2)_L)\}\end{aligned} \tag{38}$$

$$\begin{aligned}\text{class 2} = \{&((4,2)_L,(9,2)_L),((5,3)_L,(3,2)_L)\\ &((11,2)_L,(3,2)_L)\}\end{aligned} \tag{39}$$

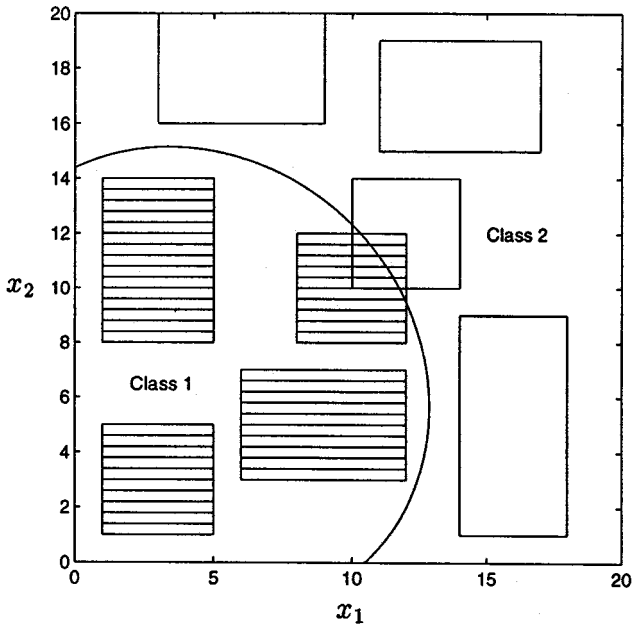

Fig. 13. The membership functions of the linguistic values "small" and "large" in Example 3.

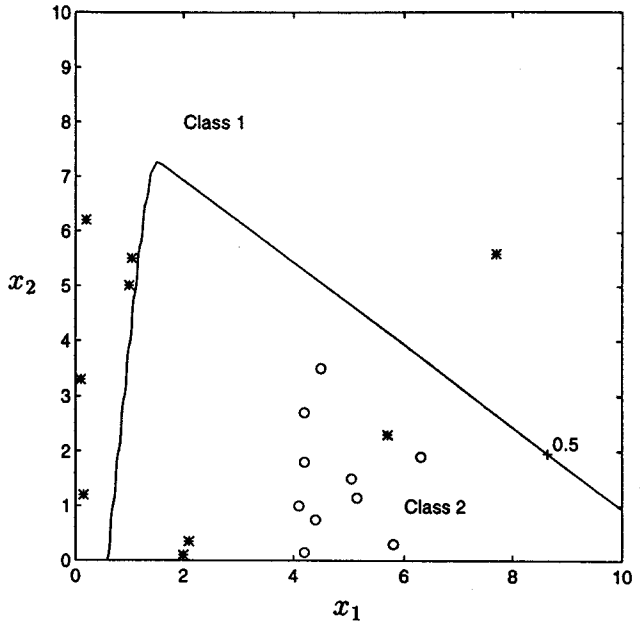

Fig. 15. The simulation result of Example 3 using the FBP algorithm.

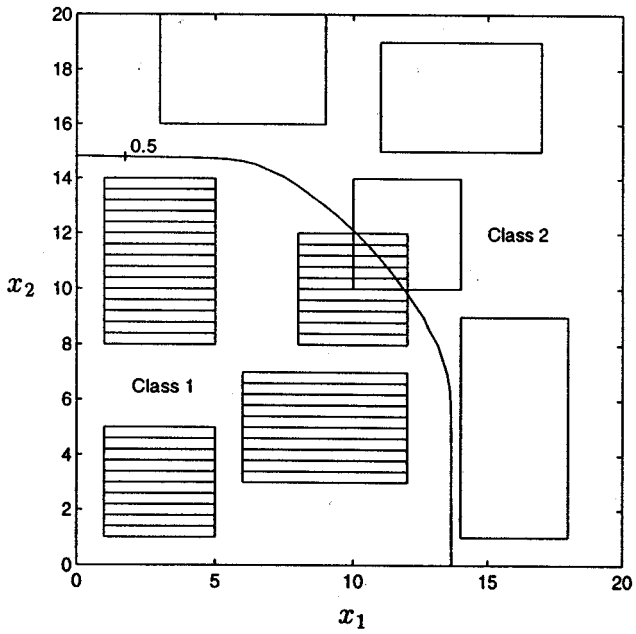

Fig. 14. The simulation result of Example 3 using the second-order fuzzy perceptron learning algorithm.

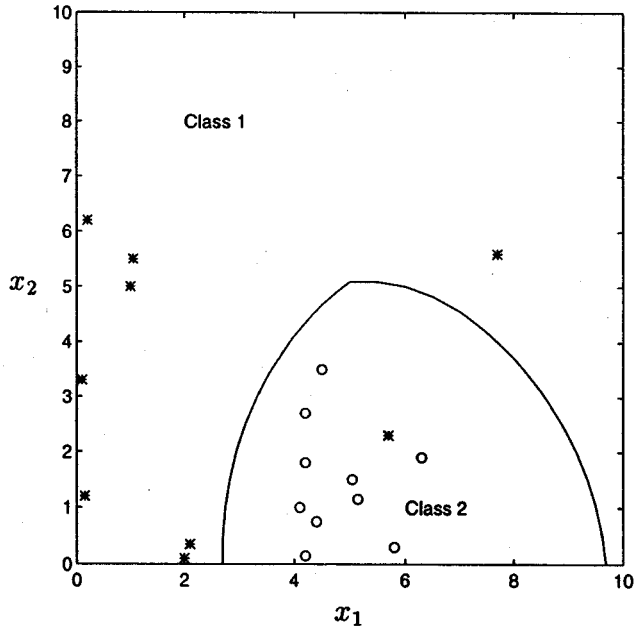

Fig. 16. A simulation result of Example 3 using the AFLC approach.

$$\text{class } 3 = \{((13,2)_L,(13,3)_L),((18,1)_L,(17,2)_L) \\ ((17,3)_L,(8,1)_L)\}. \tag{40}$$

In this example, we set $\mu = 0.1$, $\beta = 2$, and $\eta = 1.01$ for the FPNN. Fig. 17 presents the separating boundary chosen from those by the FPNN algorithm after 1000 learning epochs. The boundary curves are drawn to denote the points at which the larger two Net function values are identical. Namely, if the $\text{Net}^h_{p,1}$ value assumes the maximum value at a point, then this point is labeled as class 1. This same figure also reveals that the given nonoverlapping fuzzy vectors are correctly classified and the boundary curves crossing the overlapping areas are quite fair in minimizing the misclassified area of the overlapping fuzzy vectors. Fig. 18 displays the chosen separating boundary generated by the FBP for 1000 epochs.

The above four examples demonstrate the effectiveness and flexibility of the FPNN. The insolvability of the AFLC on Examples 2 and 4 reveals that it is not as general as FPNN and FBP for such problems. The decision boundary produced by the AFLC approach is deterministic and its design time is shorter

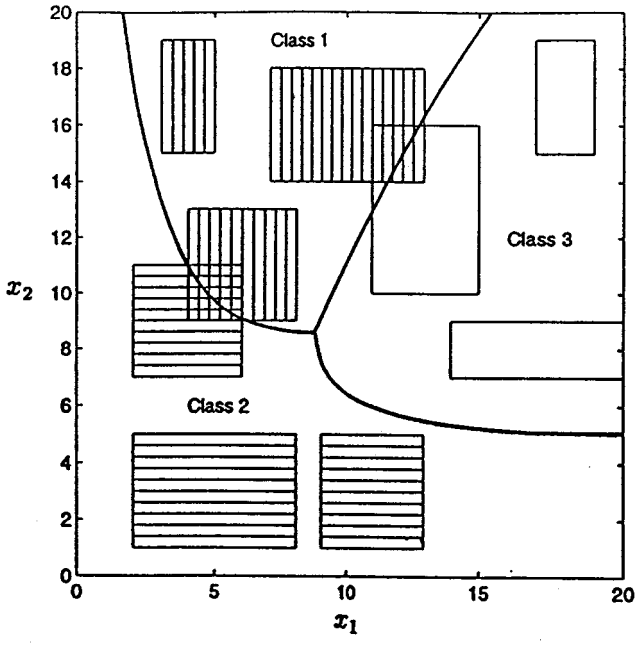

Fig. 17. The simulation result of Example 4 using the second-order fuzzy perceptron learning algorithm.

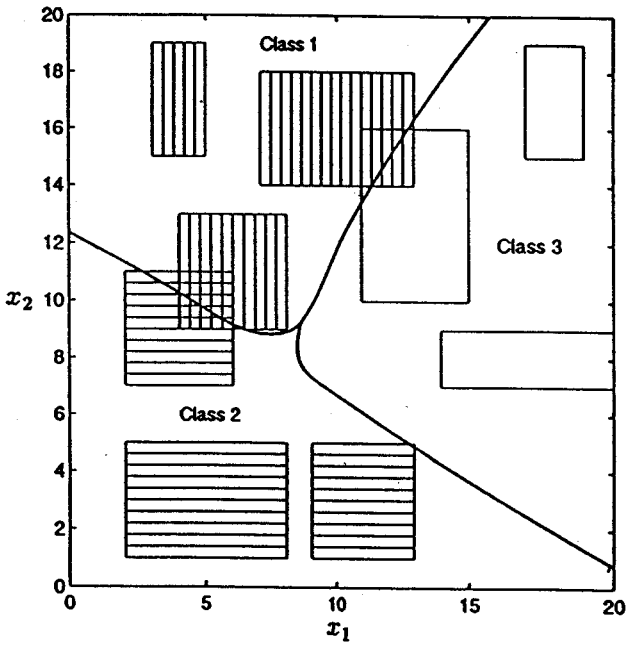

Fig. 18. The simulation result of Example 4 using the FBP approach.

than the neural-based classifier since its design phase does not include a training procedure. On the other hand, the separating boundaries produced by FPNN or FBP are not deterministic in nature; their quality of classification is best assessed by statistical measures. The consistency and quality of solutions generated by FPNN and FBP algorithms are evaluated through the learning performance statistics on these four examples. For comparison, the subsequent subsection provides statistical performance in terms of the quality of the discriminant boundary and the training times required by FPNN and FBP.

1) Recognition Rate and Speed Improvement of the FPNN Approach: As mentioned earlier, the FBP algorithm extends the backpropagation algorithm to cases involving inputs of fuzzy rules. The drawbacks of the BP algorithm such as converging to local minima and slow learning convergence still persist in this approach because it is a gradient-based technique. Previous investigations, although making some progress with respect to these defects [35]–[37], could not completely resolve these drawbacks. Moreover, determining the structure, i.e., the number of layers and hidden units in each layer, of a multilayer network is difficult and critical to a trial's success [35], [38]. To our knowledge, no previous work has successfully determined exactly how many layers and nodes the network should have, thereby avoiding situations of over learning and over fitting. Hence, in the following comparison, we set up the FBP structure according to the recommendation of [12] for Examples 1, 2, and 4.

In the statistical tests of Examples 1–4 using FBP, a satisfactory solution was not frequently obtained. As noted before, Figs. 8, 12, 15, and 18, respectively, are the best classification boundaries chosen from 200 design trials of the FBP. In our experience, whenever the final total error of a trained FBP network is roughly three times or larger than that of a satisfactory solution, the separating boundary of the classifier markedly differs from those of satisfactory ones. In the sequel, the performance of the FPNN is compared with that of the FBP approach in terms of recognition rate and the training time required. In this comparison, both algorithms were run on an HP model/712 workstation. For Examples 1, 2, and 4, FBP was performed for 1000 epochs; while for Example 3, it iterated for 12 000 epochs.

For FPNN and FBP approaches, Table I lists the average recognition rate and the average training time required over 200 design test trials with random and small initial weight vectors on these four examples, respectively. The entries of the recognition rate contain two terms. The first entry records the average crisp data recognition rate, called crisp recognition rate hereafter, and the second records the average percentage of correctly classified area of fuzzy input data, referred as fuzzy recognition rate hereafter. In Examples 2 and 4, the entries of the crisp part are missing since the training patterns are only fuzzy input vectors. The average fuzzy recognition rates on these four examples by FBP are 60.5%, 92.9%, 36.7%, and 77.2%, respectively, leading to a recognition rate of 66.8% on the average of these four examples. For the first three two-class problems, by our fuzzy perceptron neural network, the fuzzy recognition rates are all superior to those obtained by FBP algorithm. Particularly for Example 3, the fuzzy recognition rate of FPNN substantially outperforms the FBP. Regarding the multiclass task of Example 4, the proposed method also excels in terms of the amount of progression. Satisfactory results are obtained by FPNN not only for two-class tasks but also for the multiclass problem. The same outcome can be found in Table I for the crisp recognition rate comparison. For AFLC algorithm, the best fuzzy recognition rates of Examples 1 and 3 (Fig. 9 and Fig. 16) are 61.9% and 99.7%, respectively.

The training times needed for both networks were also recorded. As Table I indicates, the CPU training time required

TABLE I
THE AVERAGE TRAINING TIME RATIOS AND RECOGNITION RATES ON 200 TRIALS OF EXAMPLES 1–4

Example	FPNN Training Time(s)	FPNN Recognition Rate Crisp	FPNN Recognition Rate Fuzzy	FBP Training Time(s)	FBP Recognition Rate Crisp	FBP Recognition Rate Fuzzy	Training Time Ratio of FPNN/FBP
1	51	100%	61.5%	682	97.2%	60.5%	7.5%
2	39	—	97.6%	466	—	92.9%	8.4%
3	1990	95%	99.6%	12089	76.3%	36.7%	16.5%
4	316	—	91.2%	1100	—	77.2%	28.7%
Average		97.5%	87.5%		86.8%	66.8%	15.3%

TABLE II
MEANS AND STANDARD DEVIATIONS OF MISCLASSIFIED AREAS FOR EXAMPLES 2 AND 4 ON 200 TRIALS

Example	FPNN Mean	FPNN SD	FBP Mean	FBP SD	Ratio of FBP/FPNN Mean	Ratio of FBP/FPNN SD
2	4.20	0.16	12.55	6.07	2.98	38
4	13.10	1.11	33.79	28.82	2.57	26
Average					2.78	32

TABLE III
THE AVERAGE SUCCESSFUL CLASSIFICATION RATIOS ON 200 TRIALS OF EXAMPLES 1–4

Example	FPNN Successful Classification Ratio	FBP Successful Classification Ratio
1	100%	89%
2	100%	34%
3	100%	32%
4	100%	21%
Average	100%	44%

by FBP algorithm is significantly longer than ours. The ratios of central processing unit (CPU) training time needed for FPNN over that needed for FBP of the four examples are 7.5%, 8.4%, 16.5%, and 28.7%, respectively, and lead to a ratio of 15.3% average over these four examples.

Moreover, to quantitatively assess each scheme's solution quality, the misclassified area by the separating boundary was computed for each design test trial. The misclassified area can be employed as an evaluation index for the solution consistency of these two networks. Table II lists the means and the standard deviations (SDs) for the misclassified areas of Examples 2 and 4 over the 200 trials. This table reveals that the mean and SD obtained from FPNN are all smaller than those from FBP. In addition, averaging these two examples indicates that the mean of the misclassified areas obtained from FBP is 2.78 times larger than that achieved with FPNN. For the SD of the misclassified areas, FBP is 32 times greater than FPNN. Such large values of the means and SDs of the misclassified areas explain the solutions inconsistency observed from numerous FBP trials above, and the infrequency of the satisfactory decision boundaries after these design trials. The superiority of the misclassified areas of FPNN over FBP partially accounts for why the recognition rate of FPNN is markedly exceeds that of FBP.

To assess the reliability of the qualified convergence of FPNN and FBP, we define the successful classification trial in the following manner. For those examples with crisp and fuzzy input data, i.e., Examples 1 and 3, a test design trial is assumed to be a successful classification if all the crisp data are correctly classified. Note that the outlier training pattern $(5.7, 2.3)$ in Example 3 is neglected regardless of weather it is correctly classified or not. For those examples of only fuzzy rule input data, i.e., Examples 2 and 4, if all the nonoverlapping vectors are classified correctly, then this test design trial is considered as a successful classification. If a trial satisfies the condition described above, then this trial is labeled as a successful one. The sum of all successful trials over the total trials, i.e., 200, gives the successful classification ratio. The results in Table III indicate that the successful classification ratios on these four examples by FBP are 89%, 34%, 32%, and 21%, respectively, and lead to a successful classification ratio of 44% on the average of these four examples. By our fuzzy perceptron neural network, a 100% successful classification ratio has been obtained by averaging these four examples.

Based on the comparison above, we can conclude that FPNN can lead to a much more reliable discriminant boundary consistently than that of FBP algorithm. The proposed approach cannot only produce a very high classification rate, but also take a much shorter learning time than that by the FBP approach.

B. Simulation 2

The well-known two-spiral data set is a neural network benchmark problem for classification [34]. The training set consists of 194 points, half for each class. These training points are arranged in two interlocking spirals that go around the origin three times, as shown in Fig. 19(a) ("$\cdot$" points denote class 1 whereas "$*$" points denote class 2). Note that our FPNN is a classifier of extending single-layer structure and is capable of providing second-order discriminant functions in a distributed manner. By a divide-and-conquer strategy, we divided the two-spiral data into subregions and these subregions can be suitably dichotomized by a set of elementary forms such as paraboloids and ellipsoids provided by the FPNN models. Namely, the concept of using a number of FPNNs to cover the divided subregions was adopted in this simulation. Accordingly, we divided the two-spiral patterns into a few subsets of regions. The size of the subregion of patterns is

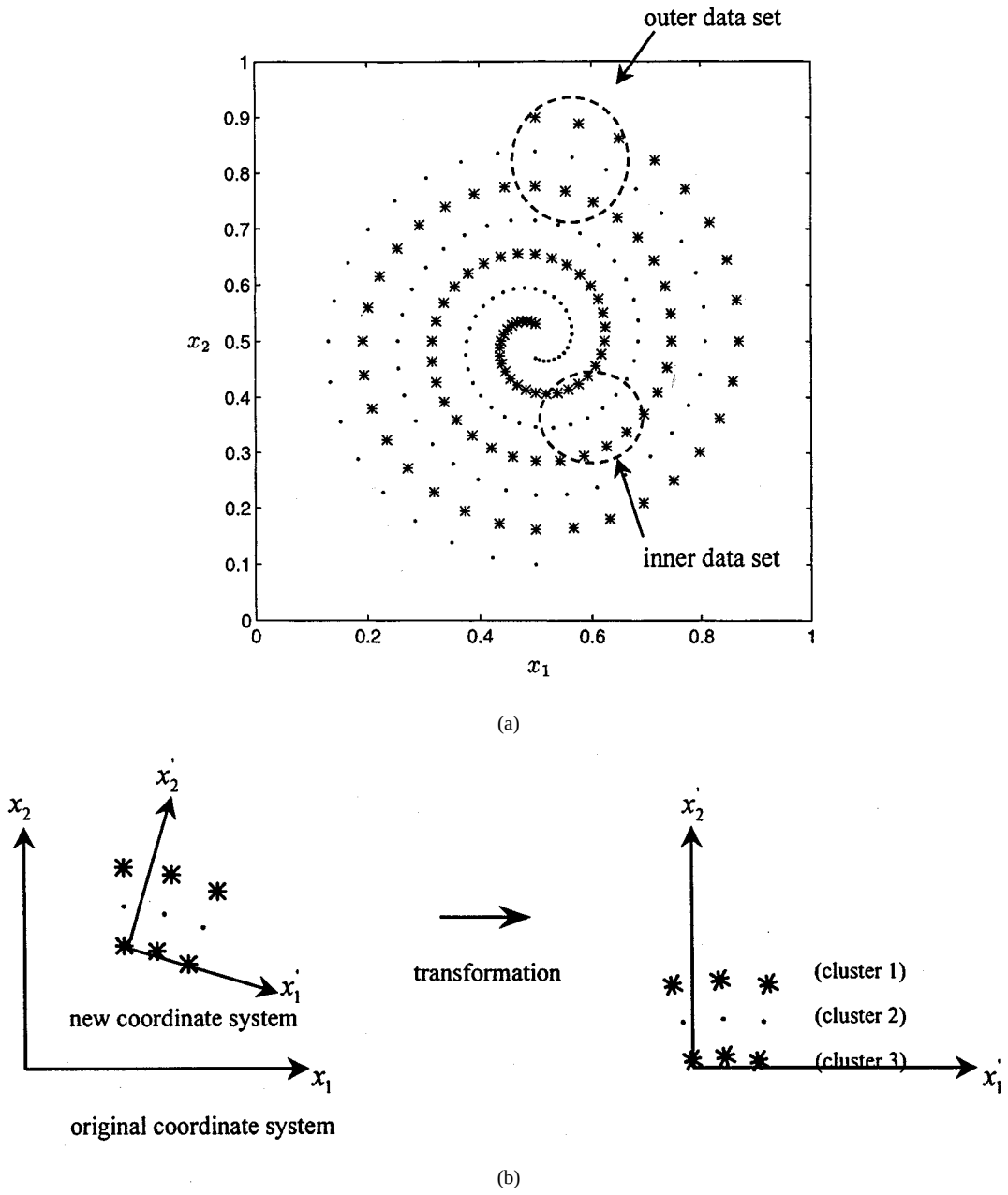

Fig. 19. (a) The training points for the two-spiral problem and the regulation of dividing the patterns into a few subsets. (b) The processes of the coordinates transformation.

somewhat inversely proportional to the patterns' density in the subregion. In this setting, the data contained in a subregion of the outer turn consists of a smaller number of patterns, while the subregion at the inner turn contains a larger number of patterns [see Fig. 19(a)] and there were 25 data subsets generated after this division and each of the 25 data subsets will be solved by an FPNN for this benchmark problem.

First, we would extract 25 sets of fuzzy IF-THEN rules, i.e., one set of rules for each data subset in a subregion. By translation and rotation techniques, a new x_1'–x_2' coordinate system was assigned to each subregion so that the patterns contained in each subregion are more easily manageable for rule extraction. Observing, for example, the first outer data subset in this new coordinates system, we can cluster these nine samples, as shown in Fig. 19(b), into three clusters through characterizing the pattern subset using IF-THEN rules concerning x_2'-coordinate feature. For instance, the three samples in the middle can be specified by a rule such as "If x_2' is medium, then it belongs to class 1." Similarly, the upper and lower three samples can be specified by the x_2'-coordinate being large and small, respectively. In this way, we use three linguistic terms, "small," "medium," and "large" of x_2'-coordinate for constructing the fuzzy rules for classification. For the definition of the membership functions, arithmetic means, of the x_2'-coordinate, of these three clusters were calculated and then used as the centers as of the corresponding symmetric triangular membership functions of (32).

The overlapping of the membership functions depends on the spread, i.e., the parameter b of (32), chosen; and in this example, the spreads of the "small," "medium," and "large" fuzzy sets of x_2'-coordinate were chosen to be three times of the standard deviations of the corresponding patterns, respectively. As to the membership function of x_1'-coordinate, because x_2'-coordinate can singly specify the characteristics of the data very effectively, membership function U of the x_1'-coordinate is chosen to be almost crisp. The minimal and maximal values of x_1'-coordinate of all patterns in the subregion (denoted as $\min_{x_1'}$ and $\max_{x_1'}$, respectively) with a small tolerance ϵ were chosen as the interval range, $[\min_{x_1'} - \epsilon, \max_{x_1'} + \epsilon]$ for full membership function and zero elsewhere. In this example, parameter ϵ was $1/20$ of the value $\max_{x_1'} - \min_{x_1'}$.

After the rule generation process above, the transformed pattern subset can be qualitatively described by the following three IF-THEN rules:

If x_1' is U and x_2' is large then it belongs to class 2
If x_1' is U and x_2' is medium then it belongs to class 1
If x_1' is U and x_2' is small then it belongs to class 2. (41)

FPNN was then used to classify the transformed samples, together with these three IF-THEN rules generated. In a similar manner, each of the other 24 data subsets was transformed to its new and suitable coordinate system and then three IF-THEN rules in the same format as (41) were also extracted. Each data subset and its corresponding fuzzy rules were trained using an FPNN model. All the parameter settings for the FPNNs were the same as in the Simulation 1 and 1000 learning epochs were taken. After all the 25 data subsets have been respectively processed by FPNNs, the 25 final decision boundaries derived by FPNNs were shown in Fig. 20. These decision boundaries were finally combined by the "OR" operator. The output is assumed to be class 1 if at least one decision boundary indicates that it belongs to class 1; otherwise, it belongs to class 2. In this figure, 25 FPNNs were used to accomplish the whole classification task and 100% recognition rate was produced.

For comparison, the same 25 data subsets and their corresponding rules were also respectively applied to the FBP algorithm. The structure and learning epochs used for the FBP were the same as those used for Example 1 of Simulation 1, and have achieved 70.48% recognition rate. The AFLC was also tested on this benchmark problem in a similar manner. Using these 25 subsets along with their corresponding IF-THEN rules as inputs, AFLC produced a high recognition rate of 97.43% under 25 sets of best-tuned parameters, σ, α, and β, obtained by trial and error. Comparing the classification accuracy obtained for the two-spiral benchmark data, the best performance still goes to the proposed FPNN approach.

V. Conclusion

To address classification problems, this paper presents a fuzzy perceptron neural network which is capable of accepting two kinds of input data: fuzzy IF-THEN rules and numerical data. Incorporating the h-level sets into the perceptron neural network effectively represents the linguistic values in fuzzy rules, thereby, the proposed neural network with fuzzy input handling ability is enhanced. At the h-level of fuzzy numbers, a fuzzy perceptron learning procedure is derived. The minimum of the fuzzy discriminant function, obtained from the modified vertex method, determines weather a fuzzy perceptron learning update step is executed or not. The derived learning algorithm extends the conventional perceptron algorithm to fuzzy input vectors. Moreover, the fuzzy pocket algorithm is derived and then further incorporated into the fuzzy perceptron learning scheme to tackle nonseparable cases. Simulation results demonstrate that the proposed algorithm not only consistently yields an accurate and efficient solution, but also resolves the limitations of inaccuracy and slow learning convergence encountered in the FBP approach.

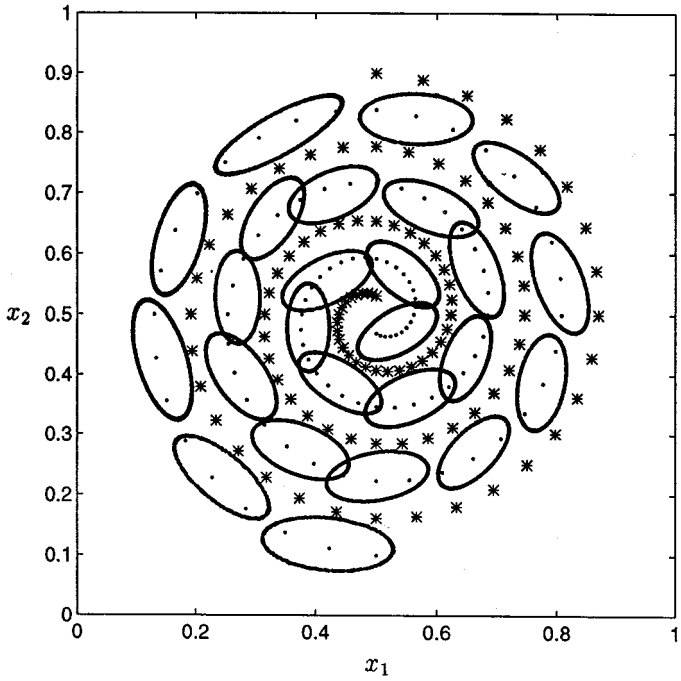

Fig. 20. The simulation result of the two-spiral problem using the FPNN approach.

Acknowledgment

The authors would like to thank the referees for their valuable comments and suggestions, which helped us to improve this paper.

References

[1] L. A. Zadeh, "Fuzzy sets," *Inform. Contr.*, vol. 8, pp. 338–353, 1965.
[2] ——, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst., Man, Cybern.*, vol. 3, pp. 28–44, Jan. 1973.
[3] ——, "The concept of a linguistic variable and its application to approximate reasoning—Parts I, II, III," *Inform. Sci.*, vol. 8, pp. 199–249, 1975. also, pp. 301–357; vol. 9, pp. 43–80.
[4] H.-J. Zimmermann, *Fuzzy Set Theory and Its Applications.* Norwell, MA: Kluwer, 1996.
[5] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems.*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[7] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*. Cambridge, MA: MIT Press, 1986, vol. 1. PDP Res. Grp..
[8] D. J. Montana and L. Davis, "Training feedforward neural network using genetic algorithms," in *Proc. 11th Int. Joint Conf. Artificial Intell.*, Detroit, MI, Aug. 1989, pp. 762–767.
[9] Y. Ichikawa and Y. Ishii, "Retaining diversity of genetic algorithms for multivariable optimization and neural network learning," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 2, Nagoya, Japan, Mar. 1993, pp. 1110–1114.
[10] W. Pedrycz, "Fuzzy sets in pattern recognition: Accomplishments and challenges," *Fuzzy Sets Syst.*, vol. 90, no. 2, pp. 171–176, 1997.
[11] W. Wei and J. M. Mendal, "A fuzzy classifier that uses both crisp samples and linguistic knowledge," in *Proc. 3rd IEEE Conf. Fuzzy Syst.*, vol. 2, Orlando, FL, June 1994, pp. 792–797.
[12] H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural networks that learn from fuzzy IF-THEN rules," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 85–97, Feb. 1993.
[13] H. M. Lee and W. T. Wang, "A neural network architecture for classification of fuzzy inputs," *Fuzzy Sets Syst.*, vol. 63, no. 2, pp. 159–173, 1994.
[14] J. L. Chen and J. Y. Chang, "Fuzzy perceptron learning and its application to classifier with numerical data and linguistic knowledge," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 6, Perth, Australia, Nov. 1995, pp. 3129–3133.
[15] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 683–697, 1992.
[16] S. Mitra and S. K. Pal, "Fuzzy multilayer perceptron, inferencing and rule generation," *IEEE Trans. Neural Networks*, vol. 6, pp. 51–63, 1995.
[17] Y. Hayashi, J. J. Buckley, and E. Czogala, "Fuzzy neural network with fuzzy signals and weights," *Int. J. Intell. Syst.*, vol. 8, pp. 527–537, 1993.
[18] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, pp. 386–408, 1958.
[19] M. Minsky and S. Papert, *Perceptrons*. Cambridge, MA: MIT Press, 1969.
[20] S. I. Gallant, "Perceptron-based learning algorithms," *IEEE Trans. Neural Networks*, vol. 1, pp. 179–191, 1990.
[21] ——, "Optimal linear discriminants," in *Proc. 8th Int. Conf. Pattern Recogn.*, 1986, pp. 849–852.
[22] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 6, pp. 693–699, 1985.
[23] K. J. Schmucker, *Fuzzy Sets, Natural Language Computations, and Risk Analysis*. Rockville, MD: Comput. Sci., 1984.
[24] D. Dubois and H. Prade, "Operations on fuzzy numbers," *Int. J. Syst. Sci.*, vol. 9, no. 6, pp. 613–626, 1978.
[25] W. M. Dong and F. S. Wong, "Fuzzy weighted averages and implementation of the extension principle," *Fuzzy Sets Syst.*, vol. 21, no. 2, pp. 183–199, 1987.
[26] G. Alefeld and J. Herzberger, *Introduction to Interval Computations.*. New York: Academic, 1983.
[27] R. E. Moore, *Interval Analysis.*. Englewood Cliffs, NJ: Prentice-Hall, 1966.
[28] J. J. Buckley and Y. Qu, "On using α-cuts to evaluate fuzzy equations," *Fuzzy Sets Syst.*, vol. 38, no. 3, pp. 309–312, 1990.
[29] R. Kruse, J. Gebhardt, and F. Klawonn, *Foundations of Fuzzy Systems.*. New York: Wiley, 1994.
[30] W. M. Dong and H. C. Shah, "Vertex method for computing functions of fuzzy variables," *Fuzzy Sets Syst.*, vol. 24, no. 1, pp. 65–78, 1987.
[31] A. Roy, L. S. Kim, and S. Mukhopadhyay, "A polynomial time algorithm for the construction and training of a class of multilayer perceptrons," *Neural Networks*, vol. 6, no. 4, pp. 535–545, 1993.
[32] G. S. Lim, M. Alder, and P. Hadingham, "Adaptive quadratic neural nets," *Pattern Recogn. Lett.*, vol. 13, no. 5, pp. 325–329, 1992.
[33] R. L. Burden and J. D. Faires, *Numerical Analysis.*. Boston, MA: PWS, 1993.
[34] K. J. Lang and M. J. Witbrock, "Learning to tell two spirals apart," in *Proc. 1988 Conf. Connectionist Models Summer School*, 1988, pp. 52–59.
[35] P. Boldi and K. Hornik, "Neural networks and principle component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, no. 1, pp. 53–58, 1989.
[36] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 76–86, 1992.
[37] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295–307, 1988.
[38] E. D. Sontag, "Feedback stabilization using two-hidden-layer nets," *IEEE Trans. Neural Networks*, vol. 3, pp. 981–990, 1992.

Jia-Lin Chen received the B.S. degree in electrical engineering from Feng-Chia University, Taichung, Taiwan, R.O.C. in 1994, and the M.S. degree in control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1996. He is currently working toward the Ph.D. degree in the Department of Electrical and Control Engineering at National Chiao Tung University, Hsinchu, Taiwan.

His current research interests include pattern recognition, fuzzy theory, and neural networks.

Jyh-Yeong Chang received the B.S. degree in control engineering in 1976 and the M.S. degree in electronic engineering in 1980, both from National Chiao Tung University, Taiwan, and the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, in 1987.

From 1976 to 1978 and 1980 to 1982, he was a Research Fellow at Chung Shan Institute of Science and Technology (CSIST), Taiwan. Since 1987, he has been an Associate Professor in the Department of Electrical and Control Engineering, National Chiao Tung University. His research interests include fuzzy sets and systems, image processing, pattern recognition, and neural network applications.