World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:6, 2008

The Labeled Classification and its Application

M. Nemissi, H. Seridi, and H. Akdag

***Abstract*—**This paper presents and evaluates a new classification method that aims to improve classifiers performances and speed up their training process. The proposed approach, called labeled classification, seeks to improve convergence of the BP (Back propagation) algorithm through the addition of an extra feature (labels) to all training examples. To classify every new example, tests will be carried out each label. The simplicity of implementation is the main advantage of this approach because no modifications are required in the training algorithms. Therefore, it can be used with others techniques of acceleration and stabilization. In this work, two models of the labeled classification are proposed: the LMLP (Labeled Multi Layered Perceptron) and the LNFC (Labeled Neuro Fuzzy Classifier). These models are tested using Iris, wine, texture and human thigh databases to evaluate their performances.

***Keywords*—**Artificial neural networks, Fusion of neural network-fuzzy systems, Learning theory, Pattern recognition.

I. Introduction

In the past few years, ANNs (Artificial Neural Networks) have been widely used in several application of the pattern recognition. They have been employed as powerful classifiers for the reason of their capacities of learning and generalizing. But among their disadvantage is the slowness of their training process. Avoiding this problem is the main goal of the labeled classification.

On the other hand, ANNs and FIS (Fuzzy Inference Systems) are appropriate to describe complex systems where it is difficult to give mathematical description. Moreover, a combination of these complementary methods allows having more robust systems [1] [2]. For example, ANNs are not interpretable, so they are not able to represent knowledge explicitly while a fuzzy system can do it by fuzzy if-then rules [1]. Furthermore, implementation of FIS necessitates tuning of the membership function parameters that can be automatically updated in the case of Neuro-Fuzzy Systems [2].

Manuscript received January 25, 2006. This work was supported by the Laboratory of Automatic and Informatics Guelma, B.P. 401 Algeria and CResTIC, Reims University, B. P. 1035, 51687, Reims Cedex 02, France.

M. Nemissi is with LAIG (Laboratory of Automatic and Informatics Guelma), Guelma University, B.P. 401, Algeria (e-mail: nemissi_m@yahoo.fr).

H. Seridi is with LAIG (Laboratory of Automatic and Informatics Guelma) Algeria and CResTIC, Reims university, B. P. 1035, 51687, Reims Cedex 02, France (e-mail: seridi@yahoo.fr).

H. Akdag is with LIP6 Université P. & M. Curie, 104, Avenue du Président Kennedy, 75016 Paris, France (e-mail: Herman.Akdag@lip6.fr).

TABLE I
ADVANTAGES AND DISADVANTAGES OF ANN AND FIS

	Advantage	Disadvantage
ANN	Parallel computing Capacity of generalization	Black box Lack of initialization techniques
FIS	Self-adaptation Possibility to use a prior knowledge	Lack of training techniques

The BP (Back Propagation) algorithm [3] is a useful algorithm in many applications. It is widely used for training the ANNs and the Neuro-Fuzzy Classifiers, but its convergence rate is relatively slow [4] and it is an unreliable algorithm [5] [6]. In order to improve performances of the BP, several approach have been proposed. Jacob et al. [7] presented an approach based on the use of a different step gains for each weight and the updating of theses term iteratively. Li et al. [8], Russo [9], Ngyen et al. [10] and Yam et al. [11] presented methods based on the weight initialization, Lee et al. [12] studied the effect of initial weight on premature saturation and Kamarthi et al. [13] introduced an algorithm based on extrapolation of each weight to accelerate the BP algorithm. Zurada [14] and Chandra et al. [15] proposed methods which are based on the adapting of the activation functions parameters. Rumelhart et al. [3] added an extra term (the momentum) and Zweiri et al. [4][16] introduced a third term (proportional factor) in addition to the learning rate and the momentum. The problem of the local minima was studied by Ampazis et al. [17], Phansalkar et al. [18] and Vitela et al. [19]. Cho et al. [20] proposed an approach based on the least-squares method to improve the BP convergence and Wang et al. [21] proposed a modified error function by adding one term to the conventional function.

The labeled classification approach proposed in this work is different from these methods in the sense that our proposition aims to improve the training process by changing the representation of examples instead of modifying the training algorithm. Indeed, this method seeks to speed up the BP convergence by adding an extra feature (labels).

Our approach can be used with several models trained by the BP, and it can be used with different techniques of acceleration and stabilization. In this work, two models of the labeled classification are proposed: LMLP and LNFC. The first model is based on the use of a neural architecture while the second is based on the use of Neuro-Fuzzy architecture. In section 2, we describe the proposed approach. In sections 3, we present the first model and we discuss its classification performances on Iris, wine, human thigh and texture databases. The second model and its performances are presented in section 4. Finally, conclusions are given in section 5.

II. The Labeled Classification

An important reason of the BP slowness is the saturation behavior of the activation function used in different layer [4]. In fact, when a sigmoid has a slope near a zero, a weight point may enter the saturation region of the weight space [5]. In such situation, the weight increment remains little even if the error is relatively large. The basic idea of our approach is to make the training example linearly separable by adding an extra feature (labels). The labels must be identical for examples belongings to the same class to ensure the linear separation and reduce the possibility of entering the saturation region. After the training process, every test example will be classified according to the classifier's outputs with all labels (Fig. 1).

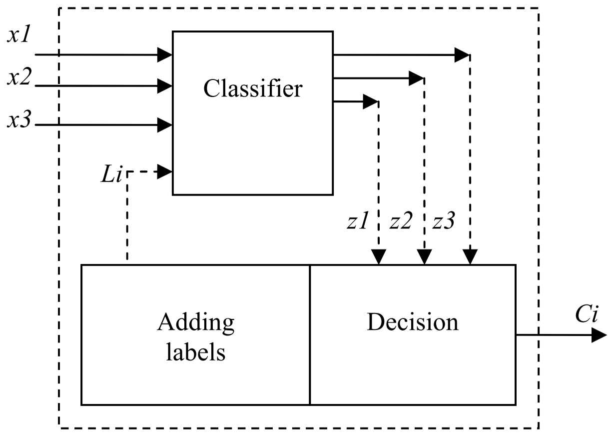

Fig. 1 Labeled classifier

Recall that a conventional classifier maps any input vector $X(x_1 x_2 \dots x_N)$ into an output vector $Z(z_1 z_2 \dots z_K)$ corresponding to the class C_i of the example represented by X. Fig. 1 shows a labeled classifier with two input and three classes. It is based on the adding of labels at the input of the used classifier and giving a decision according to the classifier outputs.

A. Methodology

The labeled classification is performed in two stages:

1. Addition of labels for all training examples and performing training of the used classifier (according to two modes).
2. Carrying out tests with these labels to classify every novel example (Fig. 2).

Therefore, for each class C_i, corresponds a label L_i. Every training example $X\ (x_1 x_2 \dots x_N)$ of C_i is represented by X $(x_1\ x_2\ \dots\ x_N\ \ L_i)$. The representations of all training examples are modified by the same manner. After the training process, every new example (X) will be tested with all labels and it will be classified according to the following decision rule:

$$X \in C_i \ \ if\ \ Er_i(X) = min\ \{Er_1(X), Er_2(X), \dots Er_K(X)\}$$

where Er_i is the sum-squared error between the target T_i corresponding to the class C_i and the calculated output Z_i using the label L_i. Er_i is defined by:

$$Er_i = \left\| T_i - Z_i \right\|$$

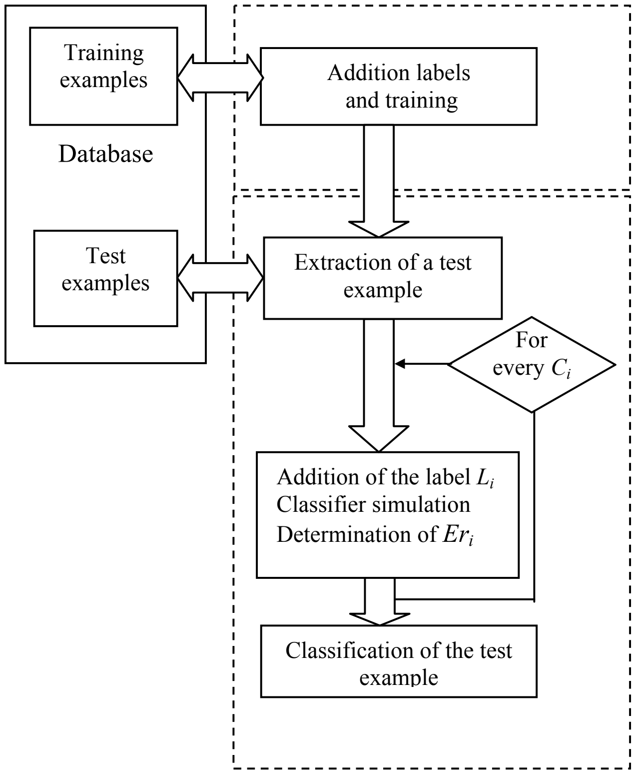

Fig. 2 Scheme of the labeled classification

B. Training in the Labeled Classification

The labeled classification is used with the classifiers trained by the BP for the reason that our decision rule is based on the sum-squared-error between target and classifier output. Training in the labeled classification is performed using two modes:

1) First Mode: Simple Training

The first mode consists in carrying out the training normally. No modifications are involved in the algorithm and the cost function is the total sum-squared-error, which is given by:

$$TSSS = \sum_{q=1}^{Q} \left\| T^{(q)} - Z^{(q)} \right\|$$

where $T^{(q)}$ and $Z^{(q)}$ are the target and the classifier output of the q^{th} example.

2) Second Mode: Full Training

In the second mode, the training is performed by minimizing the sum-squared errors between the target and the classier outputs obtained with each label (Fig. 3). That is to say, for every presented example, the classifier must be simulated and updated for all labels. The cost function becomes:

$$E = \sum_{q=1}^{Q} \sum_{i=1}^{K} \left\| T^{(q)} - Z_i^{(q)} \right\|$$

where $T^{(q)}$ is the target and $Z_i^{(q)}$ is the calculated output of the q^{th} example using the label L_i.

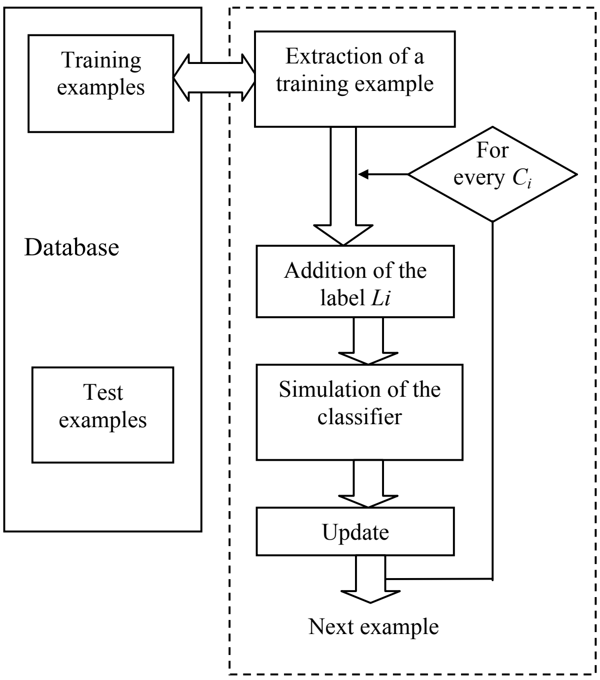

Fig. 3 Scheme of the full training

In the simple training, the adaptation is performed once at every presentation of a training example while in the full training, the adaptation is performed K times (K is the number of classes). Therefore, the process of the full training is more complex. In the next paragraphs, the performances of these two modes are evaluated.

III. The LMLP

A. Architecture

The MLP (Multi Layered Perceptron) is the most used architecture of ANN. This model provides techniques of approximating arbitrary non-linear functional mapping between multi-dimensional spaces [22]. Its training algorithm is the BP and it is important to improve performances of this process. Therefore, we propose a classification model (LMLP) based on the use of the labeled classification with the MLP. Fig. 4 represents a LMLP with N neurons at the input layer, M neurons at the hidden layer and J neurons at the output layer.

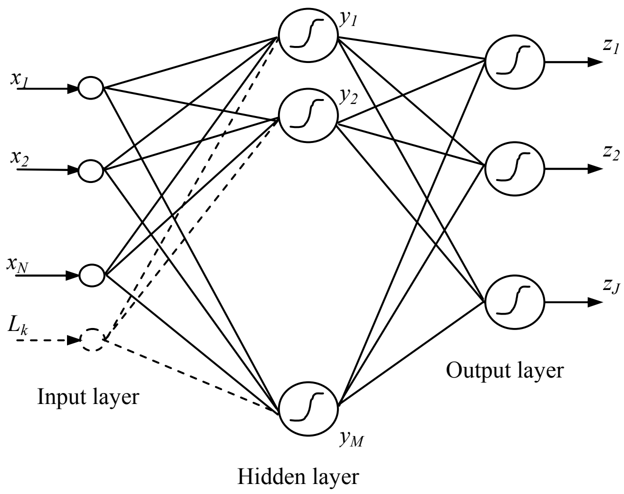

Fig. 4 LMLP

After adding labels, representation of the training example becomes: $X\,(x_1\ \ x_2\ \ldots\ x_N\ \ L_i)$ where $x_1, x_2, \ldots x_N$ are the original features and L_i the added label. This additional feature (L_i) will be treated with the same manner as the other feature. Thus, the output of the m^{th} hidden neuron will be:

$$y_m = h\left(\left\{ \sum_{n=1}^{N} x_n w_{nm} \right\} + L_i w_{N+1,m} \right)$$

The j^{th} network output becomes:

$$z_j = g\left(\sum_{m=1}^{M} h\left(\sum_{n=1}^{N} x_n w_{nm} \right) u_{mj} \right) + g\left(\sum_{m=1}^{M} h\left(L_i w_{N+1,m} \right) u_{mj} \right)$$

where z_j is the output of the j^{th} neurons, x_n is the n^{th} input. h and g are the activation function (sigmoid). The second part of the above equation shows the consequence of adding labels at the network outputs.

B. LMLP Training

1) Simple Training

In this mode, labels are treated as the other features. The LMLP is trained as any conventional MLP and the BP is used without any change. The adaptation task is to minimize the total sum-squared error (TSSE) between the classifier outputs and the targets.

2) Full Training

In this mode, for every presented example, the network outputs must be determined with all labels. The weights are adjusted according to these outputs. The function cost becomes the sum-squared errors between targets and the classier outputs obtained with different labels. Therefore, at every iteration, a training example *X(q)* is presented to the classifier with a label Li. The adaptation task is to minimize the partial sum-squared error (*PE*) between the classifier output and the target. *PE* is defined by:

$$PE = \left\| T^{(q)} - Z_i^{(q)} \right\|$$

where $T^{(q)}$ is the target and $Z_i^{(q)}$ is the calculated output of the q^{th} example using the label L_i.

C. Choice of Labels

In this process, the choice of labels is very important because they influence directly the classification performances. We suggest choosing values around *0.5* with a small deference *(δ)* between them. For example, in the case of three classes: $L_1=0.5-\delta$, $L_2=0.5$ and $L_3=0.5+\delta$

D. Iris Database Classification

To appreciate the proposed model, tests are carried out on the Iris database using an MLP and a LMLP. The classification performances of this database using MLP depend strongly on the initial weight. In some cases, the weight point enters the saturation region and the MLP makes a large number of iterations to escape from this region, or escape may never be achieved. An example of such situation is presented in Fig. 5. This figure indicates the evolution of the classification rate during the training stage of an MLP and a LMLP. Both models have the same architecture (8 hidden neurons), they are initialized by the same weights and the training parameters are the same (step gains and momentum). The used labels are $L_1=0.475$, $L_2=0.5$ and $L_3=0.525$. So, with $\delta=0.025$. LMLP1 denotes LMLP with simple training and LMLP2 denotes LMLP with full training. The graphs showed on this figure indicate the improvements obtained by the labeled classification. It allows obtaining a classification rate equal to 98 % after less than 200 iterations while the MLP permits obtaining this rate after more than 750 iterations.

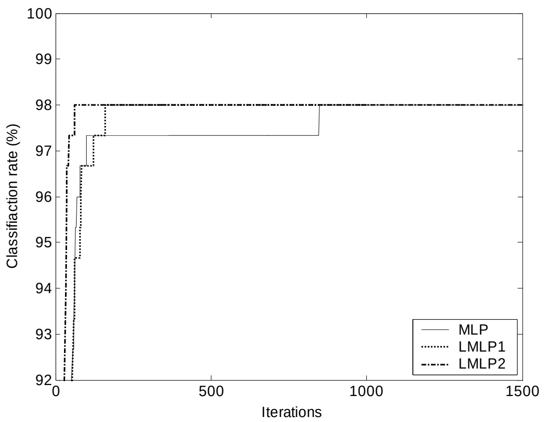

Fig. 5 Iris classification using the MLP and the LMLP

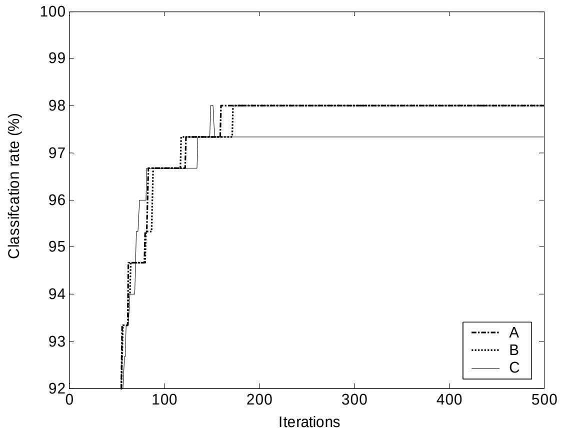

Fig. 6 Iris classification using the LMLP (simple training) with different labels

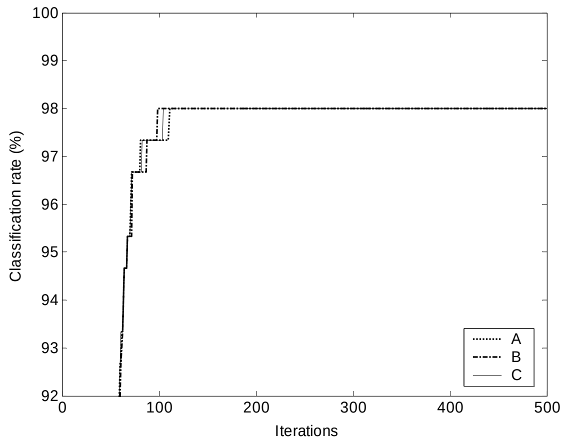

Fig. 7 Iris classification using the LMLP (full training) with different labels

Fig. 6 shows the effect of labels in the case of the simple training. Graph A corresponds to $\delta=0.025$ *(L1=0.475, L2=0.5 and L3=0.525)*, graph B corresponds to $\delta=0.015$ and graph C corresponds to $\delta=0.050$. We can note that the LMLP with simple training gives acceptable results for $\delta \leq 0.025$.

On the other hand, Fig. 7 indicates the effect of labels in the full training. It is clear that the LMLP gives the same results for these labels.

E. Wine Database Classification

This database was obtained after a chemical analysis of wines grown in the same region but derived from three different cultivars. It contains 178 examples with 13 features belonging to 3 classes. The first one has 59 examples, the second has 71 and the third has 48. We utilize the training data-itself-as-testing method.

Fig. 8 indicates the evolution of the classification rate during the training stage of an MLP and a LMLP. Both models have the same architecture (6 hidden neurons), initialization weights and parameters. The used labels are

$L_1=0.49$, $L_2=0.5$ and $L_3=0.51$. So, with $\delta=0.01$. LMLP1 denotes LMLP with simple training and LMLP2 denotes LMLP with full training. The graphs showed on this figure indicate the improvements obtained by the labeled classification (with full training). It allows obtaining a classification rate equal to 100 % after 35 iterations while the MLP permits obtaining this rate after 50 iterations.

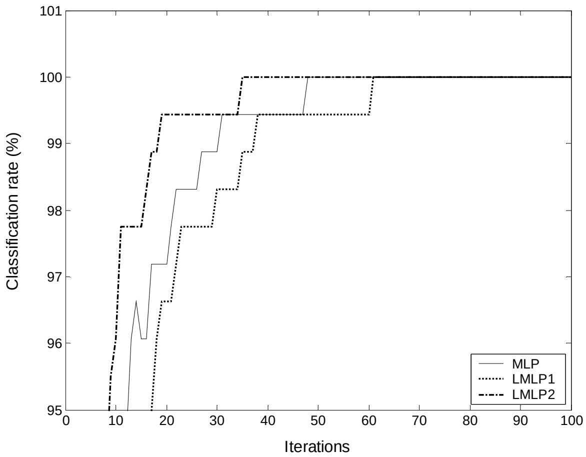

Fig. 8 Wine classification using the MLP and the LMLP

Fig. 9 shows the effect of labels in the case of the simple training. Graph A corresponds to *$\delta=0.05$ ($L1=0.45$, $L2=0.5$ and $L3=0.55$)*, graph B corresponds to $\delta=0.025$ and graph C corresponds to $\delta=0.01$. We can note that the LMLP with simple training gives acceptable results for $\delta \leq 0.025$.

Fig. 10 shows the effect of labels in the full training. The LMLP gives the same results for these different labels.

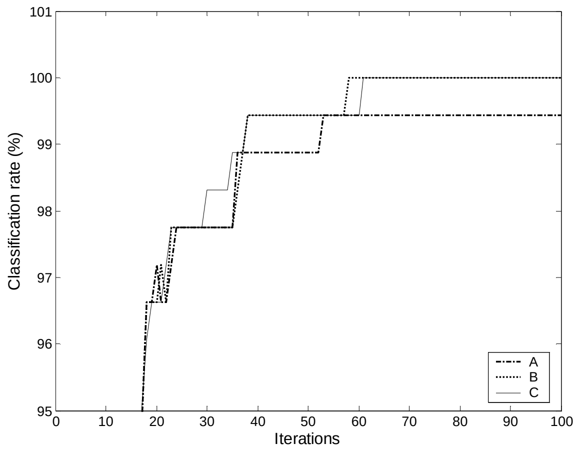

Fig. 9 Wine classification using the LMLP (simple training) with different labels

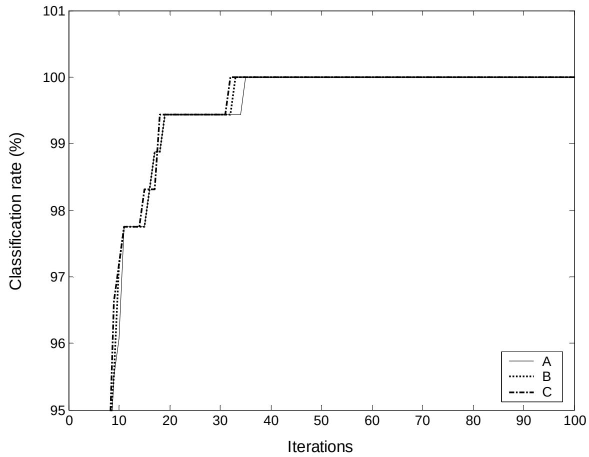

Fig. 10 Wine classification using the LMLP (full training) with different labels

F. Human Thigh Database Classification

The image of Fig. 11 is acquired by cryosection color photography. A manual classification was makes by an expert and four components were identified (grease, bone, marrow and muscle). Each one of these components corresponds to a class and a file of 300 pixels representing each one. The obtained sample consists of 1200 pixels, 300 pixels of each class. The addition of components X and Y (to locate geometrical position of a pixel and to take account of its vicinity) improves the classification performances.

To evaluate generalization capacities of our model, a cross validation of order 4 is used. Four datasets are obtained. Each base contains 900 training examples and 300 test examples.

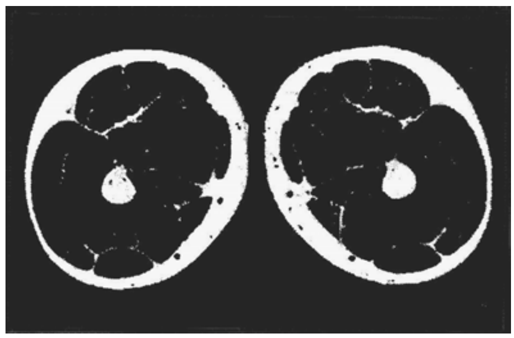

Fig. 11 Image of human thigh cryosection

As in [23], the used MLP is composed of 5 neurons at the input, 8 hidden neurons and 4 outputs neurons. The MLP and the LMLP are initialized by the same weights. In table (2), the obtained results are showed. This results are the average of the 4 data sets obtained by the cross validation.

TABLE II
RESULTS OF THE HUMANTHIGH CLASSIFICATION USING MLP AND LMLP

Classifier	Labels	Classification rate (Test datasets)
MLP		98.17
LMLP (Simple training)	0.425 0.475 0.525 0.575 (δ = 0.050)	97.83
	0.485 0.495 0.505 0.515 (δ = 0.010)	97.92
LMLP (Full training)	0.470 0.490 0.510 0.530 (δ = 0.020)	97.92
	0.485 0.495 0.505 0.515 (δ = 0.010)	97.92

G. Texture Database Classification

Fig. 12 shows an image constituted of two different microtextures. A pretreatment (calculation of different local correlations) of the initial image allows obtaining a series of 8 images. Each one is the detection result of a particular attribute. Every pixel is then described by a vector of 8 attributes.

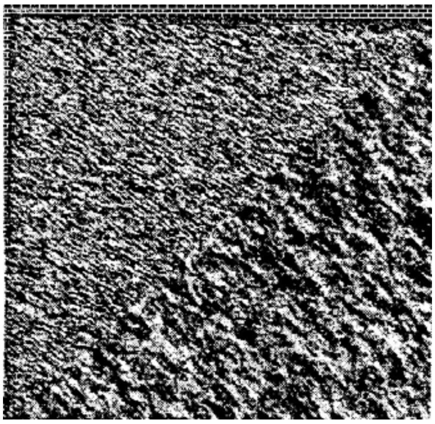

Fig. 12 Image of texture

Thus, the image is represented by two classes, which are described by eight files containing each one the pixel values. The obtained sample consists of 400 pixels of each class. A cross validation of order 4 allows obtaining four training datasets containing each one 600 pixels and 4 test datasets of 200 pixels for each one.

For the comparison, the used MLP is composed of 5 neurons at the input, 8 hidden neurons and 2 outputs neurons. Table III indicates the obtained results.

TABLE III
RESULTS OF TEXTURE CLASSIFICATION USING MLP AND LMLP

Classifier	Labels	Classification rate (Test datasets)
MLP		99.125 %
LMLP (Simple training)	0.475 0.525 (δ = 0.050)	99.375 %
	0.490 0.510 (δ = 0.020)	99.500 %
LMLP (Full training)	0.475 0.525 (δ = 0.050)	100 %
	0.490 0.510 (δ = 0.020)	100 %

IV. THE LNFC

A. Presentation

The implementation of Neuro-Fuzzy Systems aims to combine proprieties and advantages of ANNs and FIS. In these systems, every layer of ANN performs a different function of a FIS: Fuzzification, Inference and Defuzzification. The NFCs (Neuro-Fuzzy Classifiers) [1] have a Neuro-Fuzzy architecture that can incorporate in its structure fuzzy if-then rule of the form:

If x_1 is 'small' and x_2 is 'big' then X belongs to C_k

The conception of the LNFC aims to exploit and improve proprieties of the NFCs. The use of LNFC leads to replace the above rules by rules of the form [24]:

If x_1 is 'small' and x_2 is 'big' and its label is L_1 then this example belongs to C_k

B. Architecture

The labeled classification consists essentially in adding labels to all training examples. Consequently, a neuron is added at the first layer and K neurons at the second (K is the number of classes). Every neuron added to the second layer corresponds to the membership function of a label.

Fig. 13 shows an example of LNFC with two input variables (x_1 x_2) and two output variables (z_1 z_2). Every input is represented by two linguistics variables. In the first layer, every neuron corresponds to a linguistic variable. Neurons of the second layer send the product of the incoming signals and every neuron of the third layer corresponds to a class. The output of the m^{th} output of the third layer is:

$$y_m = \prod_{n=1}^{N} \mu_{nm}(x_n)$$

The j^{th} network output is:

$$z_j = \frac{\sum_{m=1}^{M} \left\{ \prod_{n=1}^{N} \mu_{nm}(x_n) \right\} w_{mj}}{\left\{ \prod_{n=1}^{N} \mu_{nm}(x_n) \right\}}$$

where μ_{nm} is the m^{th} membership function, x_n is the n^{th} input and w_{mk} is the weight between the m^{th} hidden neuron and the j^{th} output neuron. In the version of Jang [2], he used sigmoid function at the output layer. Using his model, the j^{th} network output becomes:

$$z_j = g\left(\sum_{m=1}^{M} \left\{ \prod_{n=1}^{N} \mu_{nm}(x_n) \right\} w_{mj} \right)$$

where g is the activation function of the output layer.

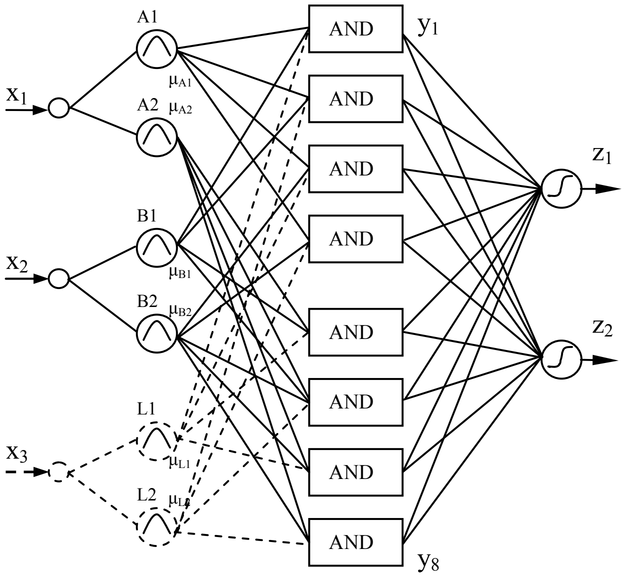

Fig. 13 LNFC with tow labels, two inputs, eight rules and two outputs

C. Training

The back propagation algorithm is widely used for neuro-fuzzy systems training, for example in [1] [24] [25] [26]. The adaptation task is to minimize the total sum-squared error between the classifier outputs and the targets. Our model is trained using this algorithm. The training of LNFC is performed without tuning the membership functions, which allows:

1. Obtaining a simplified training process because only the output weights are updated.
2. Keeping the original linguistic meaning of the membership functions.
3. Changing the T-norme operator (used in neurons of the third layer) without changing the training algorithm.
4. Changing the membership functions without changing the training algorithm.

As for the LMLP, the training can be performed using two modes: simple training and full training. In both cases, the update expression at the ($i+1$) iteration of u_{mj} is:

$$u_{mj}^{(i+1)} = u_{mj}^{(i)} + \eta_1 \left(t_j - z_j\right) g'\left(s_j\right) y_m$$

where η_1 is the step gain, t_j is the j^{th} component of the target, z_j is the j^{th} output and g' is the derivate of g (activation function of the output layer).

D. Choice of Labels

The premises of the Fuzzy rules established by the third layer depend on the membership functions of labels instead by the labels themselves. That is to say, contrary to the case of the LMLP where the choice of labels values influences directly the classification performances.

E. Iris Database Classification

To appreciate the LNFC, it is compared with a conventional NFC using Iris database. In both cases, three linguistic variables are used for the fuzzification (Fig. 14).

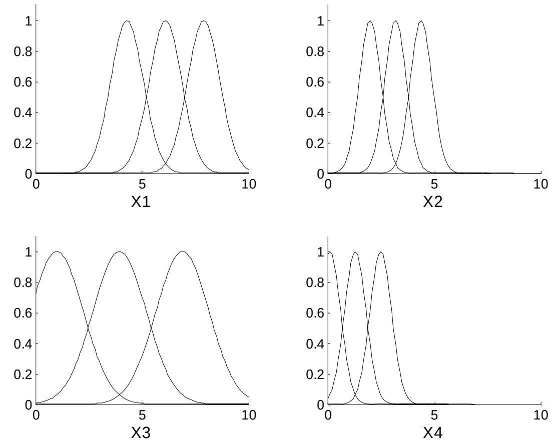

Fig. 14 Membership function of Iris features

Fig. 15 shows the evolution of the classification rate during the training stage of the NFC and the LNFC. Both models are initialized by the same weights and the training parameters are the same. The used membership function of labels are $\mu_i(Li)=1$ and $\mu_i(Lj)=0.85$. This figure indicates the improvements obtained by the labeled classification. It allows obtaining a classification rate equal to 99.33 % after 60 iterations using LNFC with full training and 98.67 using LNFC with simple training while the NFC permits obtaining this rate after 80 iterations.

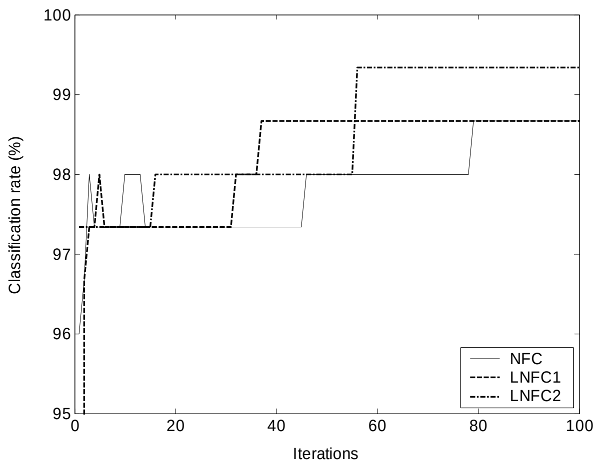

Fig. 15 Iris classification using the NFC and the LNFC

Fig. 16 shows the effect of the membership function of labels in the case of the simple training. Graph A corresponds to $\mu_i(Li)=1$ and $\mu_i(Lj)=0.9$, graph B corresponds to $\mu_i(Li)=1$ and $\mu_i(Lj)=0.8$ and graph C corresponds to $\mu_i(Li)=1$ and

$\mu_i(Lj)=0.7$. We can note that the LNFC with simple training gives an acceptable classification rate (98.67%) for $\mu_i(Lj) \geq 0.8$.

On the other hand, Fig. 17 indicates the effect of labels in the case of the full training: the LNFC allows obtaining a classification rate equal to 99.33 % for $\mu_i(Lj) \geq 0.8$.

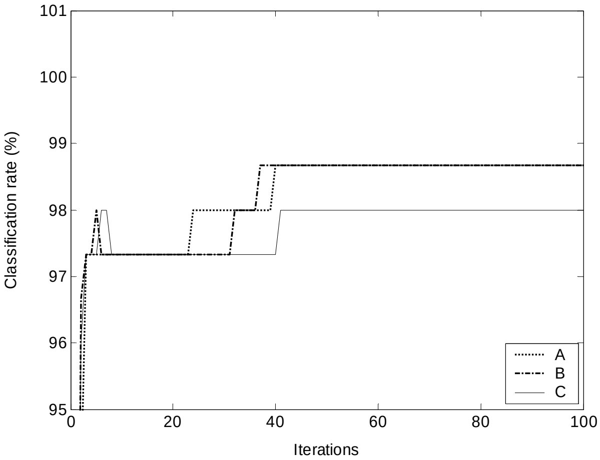

Fig. 16 Iris classification using the LNFC (simple training) with different labels

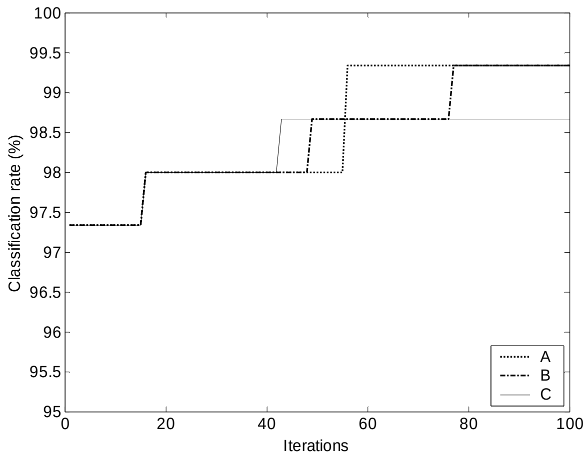

Fig. 17 Iris classification using the LNFC (full training) with different labels

F. Wine Database Classification

The LNFC is compared with NCF using wine database. In both cases, two linguistic variables are used for the fuzzification (Fig. 18).

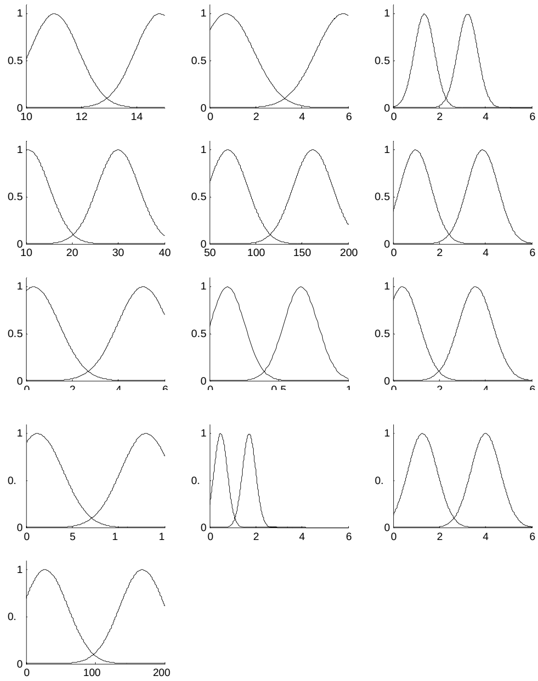

Fig. 18 Membership function of Wine features

The evolution of the classification rate during the training stage of the NFC and the LNFC are showed in Fig. 19. In both case the initial weights and the training parameters are the same. The used membership function of labels are $\mu_i(Li)=1$ and $\mu_i(Lj)=0.85$. The graphs showed on this figure indicate the improvements obtained by the labeled classification. It allows obtaining a classification rate equal to 100 % after 10 iterations using LNFC with full training and after 40 iterations using LNFC with simple training while the NFC permits obtaining this rate after 80 iterations.

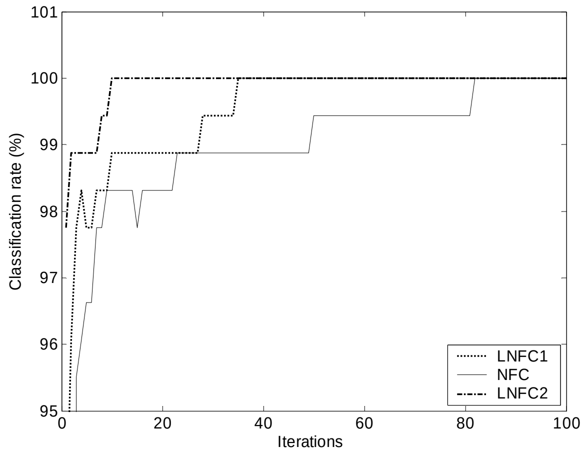

Fig. 19 Wine classification using the NFC and the LNFC

Fig. 20 shows the effect of the membership function of labels in the case of the simple training. Graph A corresponds to $\mu_i(Li)=1$ and $\mu_i(Lj)=0.9$, graph B corresponds to $\mu_i(Li)=1$ and $\mu_i(Lj)=0.8$ and graph C corresponds to $\mu_i(Li)=1$ and

$\mu_i(Lj)=0.7$. We can note that the LNFC with simple training gives acceptable results for $\mu_i(Lj)\geq 0.8$.

On the other hand, Fig. 21 indicates the effect of labels in the full training; the LNFC gives the same results for these labels.

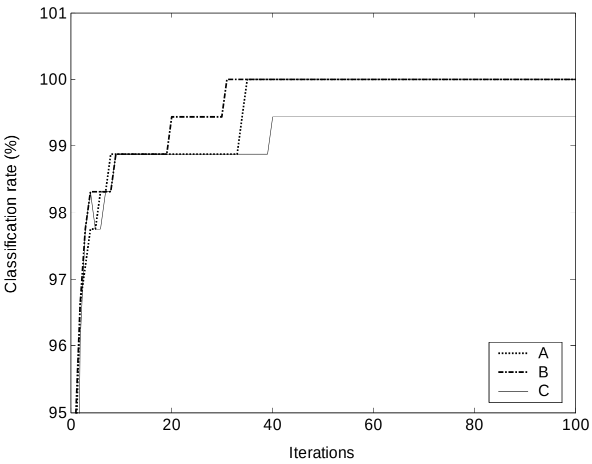

Fig. 20 Wine classification using the LNFC (simple training) with different labels

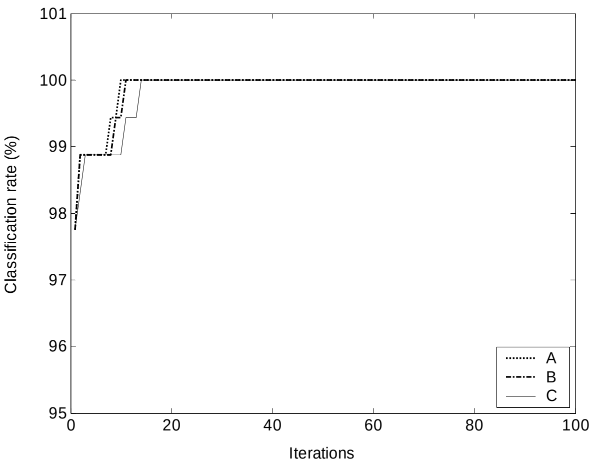

Fig. 21 Wine classification using the LNFC (full training) with different labels

G. Human Thigh Database Classification

The NFC and the LNFC are tested using human thigh database. Three linguistic variables are used for the fuzzification. Table IV illustrates the obtained results. This results are the average of the 4 data sets obtained by the cross validation.

TABLE IV
RESULTS OF THE HUMANTHIGH CLASSIFICATION USING NFC AND LNFC

Classifier	Labels	Classification rate (Test datasets)
NFC		97.92
LNFC (simple training)	$\mu i\ (Li)=1$ $\mu i\ (Lj)=0.7$	97.92
	$\mu i\ (Li)=1$ $\mu i\ (Lj)=0.9$	98.08
LNFC (full training)	$\mu i\ (Li)=1$ $\mu i\ (Lj)=0.7$	98.08
	$\mu i\ (Li)=1$ $\mu i\ (Lj)=0.9$	98.08

H. Texture Database Classification

The NFC and the LNFC are also evaluated using texture database. Two linguistic variables are used for the fuzzification. The obtained results are showed on Table V.

TABLE V
RESULTS OF THE TEXTURE CLASSIFICATION USING NFC AND LNFC

Classifier	Labels	Classification rate (Test datasets)
NFC		99.13
LNFC (simple training)	$\mu i\ (Li)=1$ $\mu i\ (Lj)=0.8$	99.25
	$\mu i\ (Li)=1$ $\mu i\ (Lj)=0.9$	99.25
LNFC (full training)	$\mu i\ (Li)=1$ $\mu i\ (Lj)=0.7$	99.25
	$\mu i\ (Li)=1$ $\mu i\ (Lj)=0.9$	99.25

V. CONCLUSION

In this paper, a new classification method is presented. Two models obtained by the use of this method are proposed: the labeled MLP and the labeled NFC. To evaluate the performances established by this development, the LMLP and the LNFC are compared respectively with conventional NFC and MLP.

Four databases are used for evaluation of these networks. Therefore, our models are examined by different type of features: length and width measure (in Iris database), pixel value and their position (in the human thigh database), local correlations components (in the texture database) and chemical features (in the wine database).

The obtained results on these databases show that the proposed approach improve performances of the MLP and the NFC except in the case of human thigh classification using MLP.

The NFC is more stable than the MLP. The LNFC provides also this property because its conception does not require any modification in the structure and the training algorithm of the NFC.

The training of the LNFC is performed without modifying the membership functions parameters, which leads obtaining simple training process and not loosing the original linguistic meaning of the membership functions. We can also change the T-norme operator (used in neurons of the third layer) and the membership functions without modifying the training algorithm.

The training in the proposed approach can be performed using two modes: the simple training and the full training.

According to our experiments, we can note that the first one is more simple but it depends strongly of labels values while the second provides more flexibility in the choice of labels but its training process is relatively complex.

REFERENCES

[1] G. J. S. Roger Jang and C. T. Sun, "A Neuro-Fuzzy Classifier and Its Applications" in *Proc. of IEEE international conference on fuzzy systems (1)*, San Francisco, 1993, pp. 94–98.

[2] B. D. Chakraborty and N. R. Pal, "A Neuro-Fuzzy Scheme for Simultaneous Feature Selection and Fuzzy Rule Based Classification" *IEEE trans., Neural Networks*. vol. 15, 2004, pp. 110-123,

[3] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representation by error propagation" *Parallel distributed processing: exploration in the microstructure of cognition*, D.E.Rumelhart and J.L.McClelland edition, MIT press Cambridge, 1986, pp. 318–362.

[4] Y. H. Zweiri, J. F. Whidborne and L. D. Seneviratne, "Three-term backpropagation algorithm" *Neurocomputing,* Vol. 50, 2003, pp. 305–318.

[5] C. Looney, *Pattern Recognition Using Neural Networks.* Oxford University Press, New York, 1997.

[6] M. A. Hoehfeld, and S. E. Fahlman, "Learning with limited numerical precision using cascade correlation algorithm" *IEEE Trans. Neural Networks* vol. 3, 1992, pp. 902-6111.

[7] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation" *Neural Networks*, vol. 1, 1988, pp. 295-307.

[8] G. li. H. Alnuweiri and W. Wu, "Acceleration of backpropagation trough initial weight pre-training with delta rule" in *Proc. 1993 IEEE Int. Conf. Neural Networks*, San Francisco, vol. 1, 1993, pp. 580-585.

[9] A. P. Russo, "Neural networks for sonar signal processing" *Tutorial No. 8, IEEE Conf. on Neural Networks for Ocean engineering*, Washington, D.C., 1991.

[10] D. Nguyen and B. Widrow, "Improving the learning speed two-layer neural networks by choosing initial values of the adaptive weights" in *Proc. IEEE Conf. Neural Networks*, San Diego, vol. 3, 1990, 21-26.

[11] J. Y. F. Yam and T. W. S. Chow, "A weight initialization method for improving training speed in feedforward neural network" *Neurocomputing*, vol. 30, 2000, pp. 219-232.

[12] Y. Lee, S. H. Oh and M. Kim, "The effect of initial weights on premature saturation in back-propagation learning" in *Proc. Joint Conf. Neural Networks*, Seattle, vol. 1, 1991, pp. 765-770.

[13] S.V. Kamarthi and S. Pittner, "Accelerating neural network training using weight extrapolations" *Neural Networks*, vol. 12, 1999, pp.1285-1299.

[14] M. Zurada, "Lambda learning rule for feedforward neural networks" in *Proc, IEEE int. Conf. Neural Networks*, vol. 3, 1993, 1808-1811.

[15] P. Chandra and Y. Singh, "An activation function adapting training algorithm for sigmoidal feedforward networks" *Neurocomputing*, vol. 61, 2004, pp. 429– 437.

[16] Y. H. Zweiri, "Optimization of a Three-Term Backpropagation Algorithm Used for Neural Network Learning" *International Journal of Computational Intelligence*. vol. 3, 2006, pp. 322–327.

[17] N. Ampazis, S. J. Perantonis and J.G. Taylor, "Dynamics of multilayer networks in the vicinity of temporary minima" Neural Networks, vol. 12, pp. 43–58, 1999.

[18] V. V. Phansalkar and P. S. Sastry, "Analysis of the back-propagation algorithm with momentum" *IEEE Transactions on Neural Networks*, vol. 5, 1994, pp. 505–506.

[19] J. Vitela and J. Reifman, "Premature Saturation in Backpropagation Networks: Mechanism and Necessary Conditions" *Neural Networks*, vol. 10, 1997, pp. 721–725.

[20] S. W. Cho and T. W. S. Chow, " Training multilayer neural networks using fast global learning algorithm-least-squares and penalized optimization methods" *Neurocomputing*, vol. 25, 1999, pp. 115–131.

[21] X.G. Wang, Z. Tang, H. Tamura and M. Ishii, "A modified error function for the backpropagation algorithm" *Neurocomputing*, vol. 57, 2004, pp. 477 – 484.

[22] C. M. Bishop, *Neural networks for pattern recognition*. Clarendon press, Oxford, 1995.

[23] M. Nemissi, H. Boudouda and H. Seridi, "Comparing performances of the MPL, RVFLNN and NFC for human thigh classification" in *Proc. of the International Workshop on Text, Image and Speech Recognition*, Annaba-Algeria, 2005, pp. 125–131.

[24] M. Nemissi, H. Seridi and H. Akdag, "Labeled Neuro-Fuzzy Classification." *Asian Journal of Information Technology*, vol. 4, 2005, pp. 868–872.

[25] F. Masulli and A.Sperduti, "Learning Techniques for Supervised Fuzzy Classifiers" *Fuzzy Learning and Applications*. CRC press, ch. 4, 2001, pp. 147–169.

[26] A. Lotfi, "Learning Fuzzy Systems" Chapter 6, *Fuzzy Learning and Applications*, CRC press, 2001, ch. 6, pp. 205–222.