Fuzzy Modification of Mixture of Experts

Abulfazl Ahmadi[*] and Mehran Rasooli [**]

[*]*Electrical Engineering Faculty, Khaje-Nasir-Toosi (KNT) University, Tehran, Iran.*

[**] *Electrical Engineering Department, Islamic Azad University, Tehran South Branch, Tehran, Iran.*

ahmadi.mail@gmail.com , Mehran.rasooli@yahoo.com

Abstract

When we are encountered with a dataset constituting imprecise clusters, usually Neural Networks (NNs) are not sufficient to classify overlapped boundaries of classes. In such a situation, fuzzy processing by which vagueness is handled sufficiently may be utilized to overcome classification difficulties. In the present paper, we use an ensemble of NNs which are trained using different subsets of entire training data set. Then a fuzzy inference unit is used to process the outputs of NNs. A criterion is introduced to modify the topologies of NNs and in addition, fuzzy rules are generated simultaneously and automatically. Also a method is presented to divide the feature space into Regions of Competence (ROC). Each classifier in the ensemble will be an expert for a ROC.

Keywords: *Pattern recognition, Classifier ensemble, Neuro-Fuzzy, Mixture of experts, Fuzzy processing*

1. Introduction

The two techniques, of neural networks and fuzzy logic are applied together for solving problems which single classical methods are not able to solve them. When combining these two techniques, we may have the merits of both. Such systems, where NNs are used to provide inputs for a fuzzy system, is preferred to be called neural/fuzzy combination [1]. Each of these tools has some advantages so that the cooperative method provides better results. In fuzzy classification, an unknown pattern belongs to all classes but with different degrees [2]. Rules and membership functions are two important parameters in a fuzzy system. Learning and generalization are the two interesting advantages of NNs which make them a serious candidate for pattern recognition tasks. Because of these capabilities, NNs are a prime candidate to combine with fuzzy systems to automate a fuzzy system. However, fuzzy tools usually implement only very restrictive learning capabilities [1]. The ability of handling uncertainties is an important feature of fuzzy systems. Also hedges may be used to modify rules in the fuzzy part of the system [3]. Always there are some ambiguities when using NNs. If we make a crisp decision on outputs of a NN, usually we tolerate some errors due to outputs closing to each other. Another disadvantage of NNs is the difficulty in determining the number of layers and number of neurons in each layer. In the proposed system, there is a criterion to check if NNs topologies are sufficient to classify patterns with a high accuracy rate, and if not, we change the topology. Trading off with two parameters of NNs, that is the number of layers and the number of neurons

is a source of complication when using NNs. One way to reduce this complication is to keep one parameter fixed (say the number of layers) and changes the number of neurons. For example if we choose a three layer NN, the number of neurons in the input and output layers is equal to the dimension of feature vectors and the number of classes, respectively. Then we can change only the number of neurons in the hidden layer.

Multiple classifier systems may generate more accurate classification than the single classifiers of the ensemble [10]. Combining classifiers are generally divided into two categories [11]: classifier selection and classifier fusion. In classifier selection each classifier is an expert in some local area of feature space. It is assumed in classifier fusion that all classifiers are trained over the whole feature space. Methods between these two categories usually are called the mixture of experts. Some methods are proposed in [11] to local competence estimates such as direct and distance-based *K*-nn estimates. In these methods it is assumed that all classifiers in the ensemble have been already trained using all training feature vectors. Once an unknown input $\boldsymbol{x}$ is received, one of classifiers is selected based on local estimates. Usually there is not a measure to check if classifiers are sufficiently designed. Another drawback is that since all classifiers are trained using all training samples, we can not expect to have classifiers that are expert in a local area of the feature space. Therefore we prefer to have classifiers which are trained using feature vectors lying in some predefined regions. Ludmila et al. [11] proposed a method based on decision templates in which outputs of all classifiers are considered.

In this paper we also propose a method in which classifiers of the ensemble are locally trained and the outputs of all classifiers are fused to classify $\boldsymbol{x}$ using a fuzzy system. Experimental results demonstrate the advantages of the proposed method (section 3).

The proposed system contains two phase: training phase and operating phase. Training phase contains two steps. In the first step of the training phase, we first employ some classifiers over entire the training data set. The topology of these classifiers may be chosen arbitrarily. Then we split clusters into at least three regions (groups), each containing vectors belonging to at least two classes. One of these regions contains classes that are classified correctly by all classifiers. Other regions contain classes in which the error rate of classification is high. Clearly clusters very close to each other constitute one region. We nominate these regions as Regions Of Competence (ROC). The next step is to train competent classifiers which are experts for each region. The average accuracy of the competent classifiers is more than the accuracy of the single best classifier which works over the whole feature space. We expect that the complexity of the competent classifiers is less than the single best classifier. In the second step of the training phase, we generate fuzzy rules. This may be performed using another set of training vectors. Generated fuzzy rules can be used as a criterion to investigate if the topologies of competent classifiers are sufficient to perform an accurate classification task (section 3). After fuzzy rules and competent classifiers are fixed, our classification system will be designed. In the operating phase, an input pattern $\boldsymbol{x}$ is presented to all classifiers in the ensemble. Outputs of all classifiers in the ensemble will be processed by the fuzzy inference unit.

Many methods have been proposed to generate fuzzy rules from a training data set [6-10]. Some methods in which neural networks are used in conjunction with fuzzy systems are proposed. In [2] the concept of membership functions has been used to define a membership matrix. Then this matrix is presented to a NN as input patterns. Indeed fuzzy rules are not considered in the classification task. In [7] a parameter ε has

been defined which is the distance between each feature and a membership function. One of the problems in mentioned methods is when patterns of high dimensionality need to be classified. If we assume patterns tolerate some redundancies [8], then a preprocessing dimensionality reduction step will discover a smaller set of features that convey all information of initial features.

The major difficulty associated with extracting fuzzy rules directly from data set is due to high dimensionality of the feature vectors. When the dimensionality of the vectors is high, many clauses will be obtained for the antecedent of the rules. One way to overcome this, is to transform the original feature vectors to a new set of vectors with lower dimensions. In fact the classifiers ensemble is considered as a transformation in which high dimensionality features is transformed to features with lower dimensions. In the present paper we frequently use the phrase 'classifier' for 'NN'.

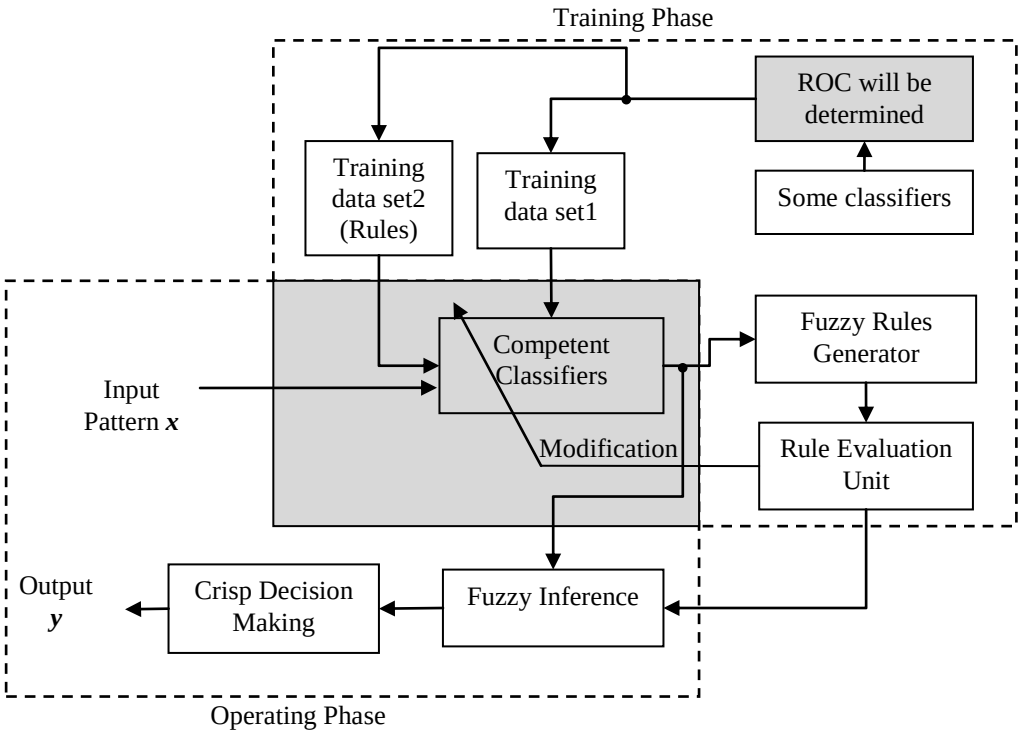

Figure 1: General block diagram of the system

2. Description of the Proposed System

Figure (1) shows the general block diagram of the proposed system. The unit "Competent Classifiers" of the system, processes input patterns using different decision functions. Each classifier in the ensemble has different discriminant function and is responsible in different regions of the d-dimensional space of the feature vectors. Here we make a soft decision instead of a crisp one at the output of the classifiers.

For the remaining of the paper suppose that:

C: The number of classes.

N: The number of classifiers in the ensemble (NNs).

K_c: The number of samples belonging to the class c in the data set which is used for generating rule.

J_i: The number of outputs of NN i.

$M = \sum_i^N J_i$: The total number of outputs of all classifiers in the ensemble.

2.1. Regions of Competence

NNs are trained through back propagation algorithm using the first group of training data set. Each NN in the ensemble is an expert for a ROC and therefore should be trained using vectors belonging to the classes lying in that region. Two initial efforts should be performed in the proposed method: 1) finding the regions of competence, and 2) finding the best classifier for each region. In some applications, it is not difficult to find clusters close to each other (see figure 6). If some priory knowledge is available about how the training samples are distributed in the feature space, we can easily choose regions of competence. For example in the problem of classifying Iris flowers, we know that the data of class 'versicolor' are very close to data of class 'virginica'. Also we can use a suitable distance measure between two clusters to find clusters close to each other. One can refer to [4] for different distance measures. However, when are going to perform a classification task with many classes, a suitable algorithm should be employed to divide entire the feature space into some regions. Such an algorithm was mentioned in section 1 and is employed in section 3.3 where our task is to classify handwriting signatures.

In the proposed system we use three layer NNs and look for the best number of neurons in the hidden layer based on the criterion discussed in section 2.3.

2.2. Fuzzy Rules

In the proposed system, J_i is in fact, the number of neurons of the output layer of NN i. Therefore we have a total of M inputs to the fuzzy block.

Fuzzy sets theory was initiated by Lotfi Zadeh [5] in 1965. Fuzzy rules are the basis of fuzzy systems and are in the form of:

if x is A **then** y is B.

Where A and B are the input and output fuzzy sets, respectively. Also x and y are input and output linguistic variables, respectively. Input features are mapped to fuzzy values by fuzzifier which uses membership functions. Fuzzy values describe the degree to which a scalar input belongs to each of the fuzzy sets. In the proposed system, we used three and two fuzzy sets for the input and output, respectively. Figures (2) and (3) show the input and output membership functions. As it can be seen in these figures, Gaussian functions are used as membership functions. As mentioned before, we use another group of training data set (except the one used to train NNs) to produce fuzzy rules.

The algorithm for generating fuzzy rules is summarized as the following steps:

1) Construct Output Matrices $OM\{c\}$. Element $O_{kj}\,; k = 1, \dots, K_c\ ,\ j = 1, \dots, M$ of each matrix $OM\{c\}$ is the jth output of the ensemble when the input $\boldsymbol{z}$ is labeled by the class $c\,; c = 1, \dots, C$.

2) Generate Rule Matrices $RM\{c\}$. Element $r_{kj}$of each matrix $RM\{c]$ is defined as:

$$r_{kj} = \begin{cases} 1 & \text{if } \mu^{\text{low}}(O_{kj}) = \max\{\mu^{\text{f}}(O_{kj})\} \\ 2 & \text{if } \mu^{\text{med}}(O_{kj}) = \max\{\mu^{\text{f}}(O_{kj})\} \\ 3 & \text{if } \mu^{\text{high}}(O_{kj}) = \max\{\mu^{\text{f}}(O_{kj})\} \end{cases}$$

In which f is one of the notations $\{\text{low}, \text{med}, \text{high}\}$.

3) Simplify $RM\{c\}$. Equal rows of matrices RM{c} should be eliminated, because they are repetitive rules. After simplification of rule matrices, the number of rows of $RM\{c\}$ will be reduced to L_c ($L_c < K_c$). In another words, L_c is the actual number of rules leading to class c.

4) Specify strong and weak rules by determining the times they have been repeated (during the last step). Rules with high repetitions are considered as strong and rules with low repetitions are weak rules.

5) Modification of rule matrices. It is possible that a weak rule in a rule matrix is a strong rule in another rule matrix and if so, we eliminate weak rules in corresponding rule matrix. However for a finite training data set, if we keep all rules, even the weaker rules, it may lead to better results. If some rules (which are often weak rules) are generated using some feature vectors affected by noise (for example due to the noise in the measuring sensors), considering them, may maximize the resulting misclassification error. Therefore a weak rule can be eliminated depending on its weakness; even it doesn't come with other rule matrices.

The number of rules produced via the above algorithm, is greater than or equal to C.

2.3. Classification of an Unknown Pattern

Now rules are at the disposal and our system is ready to classify an unknown pattern. Fuzzifying outputs of NNs for an unknown pattern will be performed for all rules. The antecedent of each rule has *M* (defined in section2) clauses and the consequent has *C* parts. Therefore rules will be the form of:

$$\prod \textbf{if } (x_m \text{ is f}) \textbf{ then } \prod (y_c \text{ is f}') \;\; ; m = 1, \dots, M \,, c = 1, \dots, C$$

Where f is defined as before and f′ is one of the notations {low, high} which indicate input and output fuzzy sets respectively. Also $\prod$ denotes the conditional **AND** operator.

If an input pattern belongs to class c, then y_c is high and $y_p \,; p = 1, \dots, C \,, p \neq c$ is low. Once a *d*-dimensional input pattern $\boldsymbol{z}$ is received, it is presented in inputs of the ensemble and a *M*-dimensional output $\boldsymbol{x} = [x_1, \dots, x_M]^T$ will be generated at the output of the ensemble. The clause "x_m is f" corresponds to the calculation of membership degree of x_m using one of membership functions depicted in figure (2). We denote the degree of membership of x_m by $\mu^{\text{f}}(x_m)$. This will yield a fuzzy value between 0 and 1. Each clause in the antecedent of a rule yields such a fuzzy value. Fuzzy intersection (which is implemented by *min*), between all fuzzy values in the antecedent of each rule, represents the firing strength of that rule.

Figure (3): output membership functions. Blue: the fuzzy set 'low' and Red: the fuzzy set 'high'.Now we construct the *C*-dimensional Antecedent Vector *AV* so that its *c*th element is equal to:

$$AV(c) = \max_{l} \left\{ \min_{j} [\mu^{f}_{RM\{c\},k,j}(x_j)] \right\}$$

$$c = 1, \dots, C \quad ; \quad l = 1, \dots, L_c \quad ; \quad j = 1, \dots, M$$

Where μ^{f} is determined by the (k, j) element of matrix $RM\{c\}$ where:

$$\mathrm{f} = \begin{cases} \text{low} & \text{if } RM\{c\}(k,j) = 1 \\ \text{med} & \text{if } RM\{c\}(k,j) = 2 \\ \text{high} & \text{if } RM\{c\}(k,j) = 3 \end{cases}$$

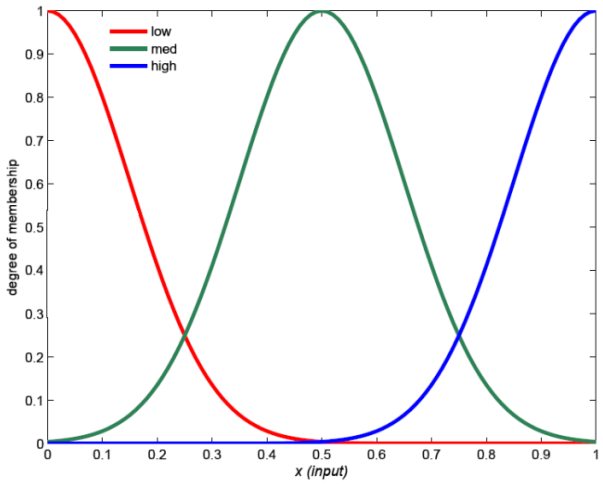

Figure 2: Input membership functions. Red: the fuzzy set "low", Green: the fuzzy set 'medium' and Blue: the fuzzy set 'high'.

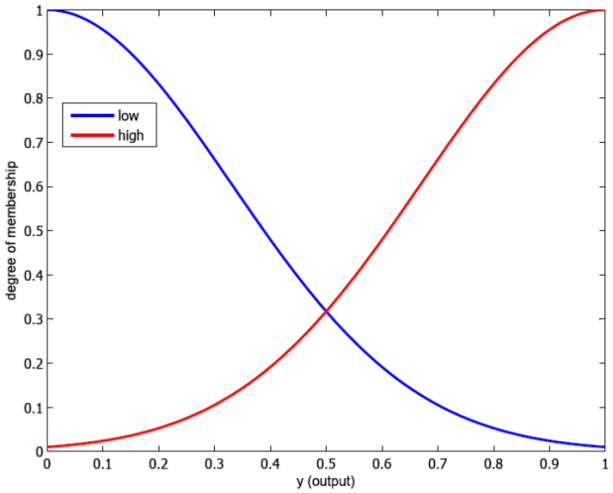

Figure 3: output membership functions. Blue: the fuzzy set 'low' and Red: the fuzzy set 'high'.

The elements of C-dimensional vector AV are denoted by $\alpha_c \;;\; c = 1, \ldots, C$. Therefore AV will become as:

$$AV = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_c \\ \vdots \\ \alpha_C \end{bmatrix}$$

The next step is the *implication* process in which we link antecedent of the rule *c* (that is α_c) to the consequents of the same rule using fuzzy operator min. Consequent in each rule has *C* parts. Here we need output membership functions shown in figure (3) that they also are Gaussian functions. The result of operator min between the scalar value α and output membership functions, which are denoted by $\mu_{low}(y)$ and $\mu_{high}(y)$ can be obtained as:

$$Y_{\alpha l}(y) = \min\left(\alpha, \mu_{low}(y)\right)$$

$$= \begin{cases} \alpha & y \le \sqrt{-2\sigma_l^2 \mathrm{Ln}\alpha} \\ \mu_{low}(y) & y > \sqrt{-2\sigma_l^2 \mathrm{Ln}\alpha} \end{cases}$$

And:

$$Y_{\alpha h}(y) = \min\left(\alpha, \mu_{high}(y)\right)$$

$$= \begin{cases} \mu_{high}(y) & y \le 1 - \sqrt{-2\sigma_h^2 \mathrm{Ln}\alpha} \\ \alpha & y > 1 - \sqrt{-2\sigma_h^2 \mathrm{Ln}\alpha} \end{cases}$$

Where σ_l^2 and σ_h^2 are variances of output membership functions of figure (3). Since the consequent of each rule has *C* parts, there are *C* outputs for each rule. In other words, the antecedent of each rule (α_c) is linked to all *C* parts of the consequent of that rule. Therefore Output Fuzzy vectors will be obtained as follow:

$$OF_{\{c\}}(y) = \begin{bmatrix} Y_{\alpha_1 l}(y) \\ Y_{\alpha_2 l}(y) \\ \vdots \\ Y_{\alpha_c h}(y) \\ \vdots \\ Y_{\alpha C l}(y) \end{bmatrix} \quad c = 1, \ldots, C$$

All elements of each vector *OF* are functions of output fuzzy variable *y*. Now corresponding elements of vectors *OF* must be combined via operator max (union of output fuzzy sets) to calculate the final fuzzy output. Final vector *T* is given as:

$$T(y)=\begin{bmatrix}\max\left\{Y_{\alpha_1 h}(y),Y_{\alpha_2 l}(y),...,Y_{\alpha_C l}(y)\right\}\\ \max\left\{Y_{\alpha_1 l}(y),Y_{\alpha_2 h}(y),...,Y_{\alpha_C l}(y)\right\}\\ \vdots\\ \max\left\{Y_{\alpha_1 l}(y),Y_{\alpha_2 l}(y),...,Y_{\alpha_C h}(y)\right\}\end{bmatrix}_{C\times 1}$$

It should ne paid attention where we have used indices *l* or *h*. It can be simply shown that the *c*th element of vector *T* is simplified to:

$$\max\left[Y_{\alpha_k l}(y),Y_{\alpha_c h}(y)\right]\ ;\ \alpha_k=\max(\alpha_j)\ ;j\neq c$$

To calculate the above expression, we have to consider two different cases for α_k and α_c : a) $\alpha_k>\alpha_c$ and b) $\alpha_k<\alpha_c$.

a) If we assume that $\alpha_k>\alpha_c$, then as demonstrated in Figure (4) we have:

$$T(y)=\max\left[Y_{\alpha_k l}(y),Y_{\alpha_c h}(y)\right]$$

$$=\begin{cases}\alpha_k & y\le\sqrt{-2\sigma_l^2\mathrm{Ln}\alpha_k}\\ \mu_{low}(y) & \sqrt{-2\sigma_l^2\mathrm{Ln}\alpha_k}<y<\sqrt{-2\sigma_l^2\mathrm{Ln}\alpha_c}\\ \alpha_c & y\ge\sqrt{-2\sigma_l^2\mathrm{Ln}\alpha_c}\end{cases}$$

b) If we assume that $\alpha_k<\alpha_c$ then as demonstrated in Figure (5) we have:

$$T(y)=\begin{cases}\alpha_k & y\le 1-\sqrt{-2\sigma_h^2\mathrm{Ln}\alpha_k}\\ \mu_{high}(y) & 1-\sqrt{-2\sigma_h^2\mathrm{Ln}\alpha_k}<y<1-\sqrt{-2\sigma_h^2\mathrm{Ln}\alpha_c}\\ \alpha_c & y\ge 1-\sqrt{-2\sigma_h^2\mathrm{Ln}\alpha_c}\end{cases}$$

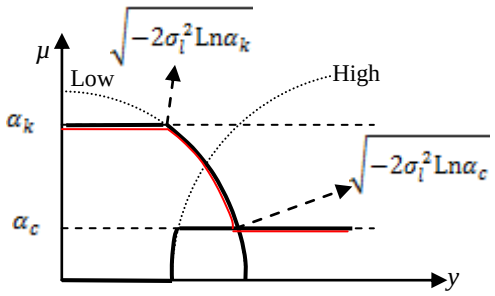

Figure 4: Calculation of Vector *T* for $\alpha_k>\alpha_c$

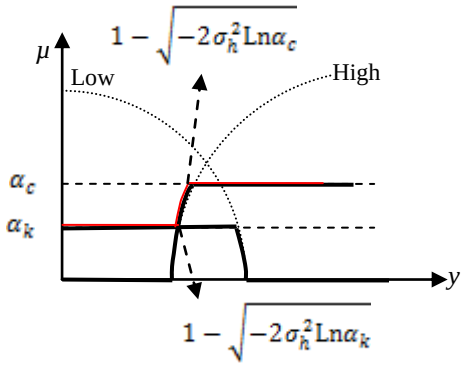

Figure 5: Calculation of Vector *T* for $\alpha_k < \alpha_c$

The vector T is still in terms of y and it needs to be defuzzified. Defuzzyfication should be performed for all elements of T. Using central gravity method, we have:

$$Dy = \frac{\sum_{y=0:\Delta}^{1} yT(y)}{\sum_{y=0:\Delta}^{1} T(y)}$$

Now decision making will be performed on scalar values of vector Dy. Although elements of T are continuous functions of output variable y, summation in the above defuzzyfication equation, can be performed on discrete values of y. Hence, we change y from 0 to 1 by steps Δ. The final class is determined by index of the largest element of Dy.

3. Evaluation of the Proposed System

Following artificial and real-world applications are used to evaluate the performance of the proposed system. In each case, results of the proposed system are compared with non fuzzy classification.

3.1. Artificial Dataset

In this section we first generate a dataset containing data from three clusters in which two clusters are too close to each other so that a single NN classifier is not able to classify data from these two clusters properly. Figure (6) shows three clusters in the 2-dimensional feature space. Each group has 60 vectors that are generated by a Gaussian distribution with $[2,3]^T$, $[7.8,7.8]^T$, $[9.5,9.5]^T$ as mean vectors and $\begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$ as covariance matrix for the three groups. We used 20 samples to train the ensemble, 20 samples to generate fuzzy rules and 20 samples to evaluate the system.

Regions of competence are also shown in figure (6). We employ two individual NNs for each ROC. Each NN is an expert in its corresponding region.

As an numerical example we suppose the ensemble contains two NNs with the topology [2 8 2] for both networks. The topology [2 8 2] means that a 3-layer NN which has 2, 8 and 2 neurons in its input, hidden and output layers, respectively.

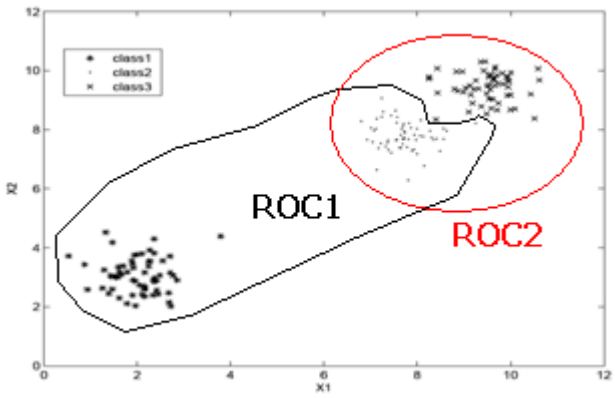

Figure 6: A Dataset Containing Three Clusters

The matrices *RM* which will be generated for a dataset containing three clusters using mentioned procedure are:

$RM\{1\} = (3 \quad 1 \quad 3 \quad 1)$

$RM\{2\} = (1 \quad 3 \quad 3 \quad 1)$

$RM\{3\} = \begin{pmatrix} 1 & 3 & 3 & 1 \\ 1 & 3 & 2 & 2 \end{pmatrix}$

For example the row $(1 \quad 3 \quad 3 \quad 1)$ in matrix $RM\{2\}$ corresponds to the following rule:

If x_1 is *low* **AND** x_2 is *high* **AND** x_3 is *high* **AND** x_4 is *low* **then** the class is 2.

In which x_i; i=1,..,4 are outputs of the ensemble.

During the generation of rule matrices *RM*, Also we calculate the repetition of each rule. In this example the value of repetition for rules introduced by both $RM\{2\}$ and $RM\{1\}$ has been calculated equal to 20. But this parameter for $RM\{3\}$ is equal to (15,5). It means that the first and the second rules have been repeated 15 and 5 times respectively. The stronger rule of $RM\{3\}$ (that is $(1 \quad 3 \quad 3 \quad 1)$) is identical to the single rule of $RM\{2\}$, but this rule in the later rule matrix is strong as much as possible (that is 20). The second rule of $RM\{3\}$ is not as strong as the first one, and on the other hand, we probably have to eliminate the first rule from rule matrix $\{3\}$. As a result the chosen topologies of NNs are not sufficient for an appropriate classification.

Now we decrease the number of neurons in the hidden layer of one of two NNs to 2 neurons (the one that is an expert in ROC1). Then we will obtain new rule matrices as:

$RM\{1\} = (3 \quad 1 \quad 3 \quad 1)$; repetition $= 20$

$RM\{2\} = (1 \quad 3 \quad 3 \quad 1)$; repetition $= 20$

$RM\{3\} = \begin{pmatrix} 1 & 3 & 1 & 3 \\ 1 & 3 & 2 & 2 \\ 1 & 3 & 3 & 1 \end{pmatrix}$; repetition $= (12,5,3)$

Looking at these matrices, the conclusion we reach is that the third rule in $RM\{3\}$, not only is a weak rule but also has been repeated in $RM\{2\}$ which is a strong rule. Thus, for a better classification we will eliminate the third rule in rule matrix $\{3\}$. With new topology of NNs, we have generated a new rule $(1 \; 3 \; 1 \; 3)$ which will improve the classification task.

Table (1) shows the classification results of dataset shown in figure (5) using a single NN with different topologies, crisp analysis of parallel NNs and fuzzy analysis of same parallel NNs.

Table 1: Classification Results of Data Set shown in Figure (5)

Method	Classification rate (%)	Description
Single NN	67 (max)	Best topology
Crisp analysis of NNs	83	Two parallel NNs
Fuzzy analysis of NNs	95	Same two parallel NNs

3.2. Iris Data Classification Problem

In recent years, many methods have been proposed to handle the Iris data (fisher 1936) classification problem [6]. Chen et al. [6] proposed a method based on fuzzy rules in which they used some threshold values defined by the user. They also presented a table where some methods previously proposed were compared with their method and it is given again in table (3). Also results of Iris data classification using the proposed method are given in this table. Simulations show that all test samples in the Iris dataset were classified by the proposed method. For the task at hand, we used 20 samples to train 2 the NNs in the ensemble, 20 samples to generate fuzzy rules and the rest of the samples in the dataset were used to evaluate the system. Topologies for the NNs are chosen as [4 8 2] to be trained using samples in clusters setosa and versicolor (NN1), and [4 12 2] to be trained using samples in clusters versicolor and virginica (NN2). Rule matrices for Iris data classification are generated as:

$RM\{1\} = (3\ \ 1\ \ 3\ \ 1)$, repetition = 20

$$RM\{2\} = \begin{pmatrix} 1 & 3 & 1 & 3 \\ 1 & 3 & 3 & 1 \\ 1 & 3 & 2 & 2 \end{pmatrix} \text{ , repetiotion } = \begin{pmatrix} 2 \\ 16 \\ 2 \end{pmatrix}$$

$$RM\{3\} = \begin{pmatrix} 1 & 3 & 1 & 3 \\ 1 & 3 & 2 & 2 \end{pmatrix} \text{ , repetition } = \begin{pmatrix} 18 \\ 2 \end{pmatrix}$$

As can be seen from rule matrices, rules (1 3 1 3) and (1 3 2 2) are repeated in both $RM\{2\}$ and $RM\{3\}$. The first rule has been repeated 18 times in rule matrix {3} but only 2 times in rule matrix {2}, therefore this rule is a weak rule for classifying an unknown sample that is labeled by class 2 and can be eliminated from rule matrix {2}. The rule (1 3 2 2) is a weak rule for both rule matrices {2} and {3} and since the strength of this rule is the same for both rule matrices, it can be kept or eliminated (here we eliminate this rule from both rule matrices).

The final rules for Iris data classification will be as:

$RM\{1\} = (3 \quad 1 \quad 3 \quad 1)$

This corresponds to the rule:

If x_1 is *high* **AND** x_2 is *low* **AND** x_3 is *high* **AND** x_4 is *low* **then** the flower is *setosa*.
And as before x_i i=1,…,4 are outputs of NN1 and NN2.

$RM\{2\} = (1 \quad 3 \quad 3 \quad 1)$

This corresponds to the rule:

If x_1 is *low* **AND** x_2 is *high* **AND** x_3 is *high* **AND** x_4 is *low* **then** the flower is *versicolor*.

$RM\{3\} = (1 \quad 3 \quad 1 \quad 3)$

This corresponds to the rule:

If x_1 is *low* **AND** x_2 is *high* **AND** x_3 is *low* **AND** x_4 is *high* **then** the flower is *virginica*.

To the inputs of the two NNs are Sepal Length (SP), Sepal Width (SW), Petal Length (PL) and Petal Width (PW) as coordinates of feature vectors.

The performance of the proposed method will be evident if we consider only PL and PW of Iris flowers as 2-dimensional feature vectors instead of 4-dimensional vectors. Figure (7) shows the corresponding 2-dimensional feature space. For the problem at hand, we use the same topologies as section 3.1 for NNs in the ensemble, that is [2 8 2] for the network1 and [2 2 2] for the network2. As given in table (2), crisp processing of the ensemble shows a correct classification rate of 80% when using 2-dimension vectors. By the proposed method, rules are generated as follow:

$RM\{1\} = (3\ 1\ 3\ 1)$, repetition $= 20$

$RM\{2\} = (1 \quad 3 \quad 3 \quad 1)$, repetiotion $= (20)$

$$RM\{3\} = \begin{pmatrix} 1 & 3 & 1 & 3 \\ 1 & 3 & 2 & 2 \\ 1 & 3 & 3 & 1 \end{pmatrix} \text{ , repetition} = \begin{pmatrix} 8 \\ 2 \\ 10 \end{pmatrix}$$

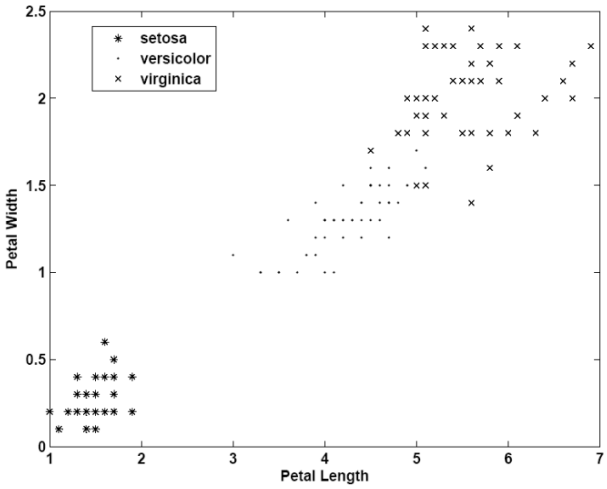

Figure 7: 2-dimensional features of Iris data. Only PL and PW are considered as features.

Table 2: Classification results of Iris data set. PL and PW were used to construct 2-dimensional feature vectors

Method	Classification rate (%)	Description
Crisp analysis of parallel NNs	91	4-dimensional feature vectors
Fuzzy analysis of parallel NNs	100	4-dimensional feature vectors
Crisp analysis of parallel NNs	83	2-dimensional feature vectors
Fuzzy analysis of parallel NNs	97	2-dimensional feature vectors

Although the rule $(1 \quad 3 \quad 3 \quad 1)$ is common for both $RM\{2\}$ and $RM\{3\}$, simulations show that if we keep it in both rule matrices, we will get better results because $RM\{3\}$ has another strong rule (that is $(1 \quad 3 \quad 1 \quad 3)$) that can causes the improvement of classification task. The rule $(1 \quad 3 \quad 2 \quad 2)$ may be considered as a weak rule but it appears only in rule matrix $\{2\}$ and we prefer not to eliminate it. Simulations show a correct classification rate of 97% using the proposed method with the above rule matrices (in the case of 2-dimensional feature vectors).

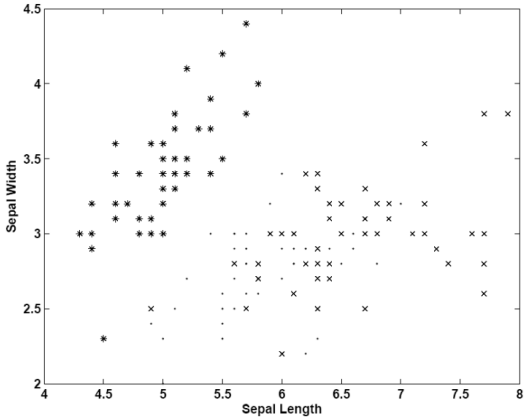

Figure 8: 2-dimensional features of Iris data. Only SL and SW are considered as features.

As another example in this section we use only SL and SW of flowers as 2-dimensional feature vectors. This is the case that rules are not sufficient to perform Iris flowers classification task. Figure (8) shows these vectors in 2-dimensional feature space. As can be seen in this figure, classes 'Versicolor' and 'Viginica' are approximately merged into a single cluster with some vectors appearing as outliers.

Rules when using SL and SW as features will be generated as:

$$RM\{1\} = (3 \;\; 1 \;\; 2 \;\; 2) \;, \;\; \text{repetition} = 20$$

$$RM\{2\} = \begin{pmatrix} 1 & 3 & 2 & 2 \\ 1 & 3 & 3 & 1 \\ 2 & 2 & 2 & 2 \end{pmatrix} \;, \;\; \text{repetiotion} = \begin{pmatrix} 17 \\ 2 \\ 1 \end{pmatrix}$$

$$RM\{3\} = (1 \quad 3 \quad 2 \quad 2) \;, \;\; \text{repetition} = 20$$

The rule $(1 \quad 3 \quad 2 \quad 2)$ is common to both rule matrices {2} and {3}. This is the only rule possessed by $RM\{3\}$. Also $RM\{2\}$ doesn't have another strong rule that could discriminate patterns between classes {2} and {3}.

Table 3: A comparison of the average classification accuracy rates of different methods for Iris data classification [6].

Methods	Average classification accuracy rate (%)
Hong and Lee's method (1996) (training data set:75 instances; testing data set:75 instances)	95.57
Hong and Chen's method (1999) (training data set:75 instances; testing data set:75 instances)	95.57
Castro's method (1999) (training data set:120 instances; testing data set:30 instances)	96.60
Chen and Fang's method (2005a) (training data set:120 instances; testing data set:30 instances)	96.72
Chen and Tsai's method (2005) (training data set:120 instances; testing data set:30 instances)	96.82
Chen and Chang's method (2005) (training data set:120 instances; testing data set:30 instances)	96.88
Chen and Fang's method (2005b) (training data set:120 instances; testing data set:30 instances)	96.96
Chen and Tsai's method (20007) [6] (training data set:120 instances; testing data set:30 instances)	97.166
The proposed method (training data set:120 instances; testing data set:30 instances, 4-dimensional features)	100
The proposed method (training data set:120 instances; testing data set:30 instances, 2-dimensional features)	97

3.3. Handwriting Signature Recognition

Signatures are special case of handwritings. Many methods [12-13] are proposed for the task of signature recognition in which some fusion of features was used to classify handwriting signatures. In this section, we are not going to focus on some fusion of features to reach a high classification rate, but instead we aim at comparing conventional classifier combining methods with the proposed method in which we use only a simple feature. We first obtain the thinned version of the signature image and then divide it into 5×5 sub images. Then we calculate normalized concentration of foreground pixels in each sub image. As a result we extract a 25-dimensional feature vector from each signature image (as shown in figure 9). Among 375 signatures (25 classes and 15 signatures in each class) of our bank, 125 signatures were used to train three different NNs that are used to find regions of competence (first step of the training phase). This is the algorithm we use to find regions. By investigating the results of the three NNs using a simple algorithm, we divide all signature classes into three regions: one region containing all signature classes that are classified correctly by the three NNs and the other two regions include signature classes that are misclassified. For the best selection of classes for constituting the two last regions, we assign classes which are misclassified for each other to a single region. For example we assign classes 11 and 5 to the same region if at least one of the three NNs classifies a signature as class 11 while it really belongs to class 5 or vice versa. If an incorrect classification is repeated in all classifiers, it will be considered as a strong misclassification.

$$\begin{pmatrix} 0.00 & 0.00 & 1.00 & 1.07 & 0.00 \\ 0.59 & 0.70 & 2.51 & 3.07 & 1.44 \\ 2.74 & 1.11 & 1.14 & 3.11 & 1.88 \\ 1.66 & 1.62 & 4.29 & 3.92 & 2.81 \\ 1.44 & 5.00 & 0.29 & 0.14 & 0.00 \end{pmatrix}$$

(a) (b) (c)

Figure 9: extracting a feature vector from a signature image. a) A binary version of signature image b) image (a) after thinning and being divided into 5×5 grids c) resulting feature vector

After presenting all training samples to the predesigned NNs, we may categorize all classes in three different regions. All NNs will correctly label some classes at the same time. Therefore we assign these classes to region (1). Other classes are those that are incorrectly classified by at least one of NNs. We assign these classes to regions (2) and (3). The topologies of the three mentioned 3-layer NNs are [25 100 25], [25 66 25] and [25 25 25] with the classification accuracy rate of 89%, 84% and 82% respectively.

The three regions obtained using the procedure mentioned above, will contain 7, 8 and 10 signature classes respectively.

Now we design three classifiers with lower complexities which are expert for each region. The classifiers selected to classify signature classes in each region are 3-layer [25 25 7], [25 25 8] and [25 25 10] NNs. The amount of classification accuracy rate for each expert is 98%, 91.5% and 87.5% respectively. As shown in table (4), with a crisp processing of the three experts, the classification accuracy rate will be improve to $\approx 91\%$ and a classification accuracy rate of $\approx 95.2\%$ will be obtained with fuzzy processing of experts. In this application, the same training set was used for both training the ensemble and generating rules.

4. Conclusion

We proposed a system containing a classifier ensemble in conjunction with a fuzzy unit to improve the classification task. Also the performance of the proposed method was illustrated by the use of some practical applications such as Iris data classification and handwriting signatures recognition. Improvement of the system is due to employing both competent classifiers which are experts in corresponding regions and fuzzy analysis of the ensemble. A considerable improvement has been achieved by a combination of neural networks and fuzzy reasoning, especially in Iris data classification problem and also signature classification task.

Table 4: Signature recognition accuracy rate using a simple feature extraction scheme.

Method	Classification rate (%)	Description
The best single classifier	89	4-dimensional feature vectors
Crisp analysis of experts	91	4-dimensional feature vectors
Fuzzy analysis of experts	95.2	2-dimensional feature vectors

References

[1] Detlef Nauck, "*Neuro-Fuzzy Systems: Review and Prospects*", Fifth European Congress on Intelligent Techniques and Soft Computing, Aachen, sep. 8-11, 1997, pp. 1044-1053.

[2] Ashis Ghosh, B. Uma Shankar,Saroj K. Meher, "*A novel approach to neuro-fuzzy classification* ", Neural Networks 22 (2009) 100-109.

[3] Bayram Cetisli, "Development *of an adaptive neuro-fuzzy classifier using linguistic hedges*", Expert systems with Applications, 37 (2010) 6093-6101.

[4] Sergios Theodoridis, Konstantinos Koutrumbas, "*Pattern Recognition*" ELSEVIER ACADEMIC PRESS.

[5] L.A. Zadeh, "Fuzzy Sets", Information and control, 1965, Vol. 8, pp. 338-353.

[6] Shyi-Ming Chen, Fu-Ming Tsai, "*Generating fuzzy rules from instances for fuzzy classification systems*", Expert Systems with Applications 35 (2008) 611-621

[7] Tzuu-Hseng S.Li, Nai Ren Guo, Chao-Lin Kuo, "*Design of adaptive fuzzy model for classification problem*", Engineering Application of Artificial Intelligence 18 (2005) 297-306

[8] Qiang Shen, Alexios Chouchoulas, "*A rough-fuzzy approach for generating classification rules*", Pattern Recognition 35 (2002) 2425-2438

[9] Sushmita Mitra, Yoichi Hayashi, "*Neuro-Fuzzy Rule Generation*, IEEE Transaction On Neural Networks, Vol. 11 No. 3 (2000)

[10] Ludmila I. Kuncheva, James C.Bezdek, Robert P.W Duin, "*Decision templates for multiple classifier fusion: an experimental comparison*" Pattern Recognition 34 (2001) 299-314

[11] K. Woods, W.P. Kegelmeyer, K. Bowyer, "*combination of multiple classifiers using local accuracy estimates*", IEEE Trans. Pattern Anal. Match. Intell. 19 (1997) 405-410

[12] M.A. Ismail, Samia Gad, "*Off-line Arabic signature recognition and verification*", Pattern Recognition 33(2000) 1727-1740

Authors

A. Ahmadi got a BSc degree in electrical engineering from KNT University of technology and a MSc degree in electrical engineering from Azad university. Currently he is working as a researcher and laboratory teacher for KNT University. He has been teaching in some universities in Iran suc h as Azad University, Shahid Rajaei University and Payam Noor University since 1999.

M. Rasooli got his BSc and MSc degrees in electrical engineering from Azad Unive rsity (Iran). Her research topics of interest are image processing and pattern recognition.