



# Learning hybrid neuro-fuzzy classifier models from data: to combine or not to combine?<sup>☆</sup>

Bogdan Gabrys\*

*School of Design, Engineering & Computing, Computational Intelligence Research Group, Bournemouth University,  
Poole House, Talbot Campus, Fern Barrow, Poole BH12 5BB, UK*

---

## Abstract

To combine or not to combine? This very important question is examined in this paper in the context of a hybrid neuro-fuzzy pattern classifier design process. A general fuzzy min–max neural network with its basic learning procedure is used within five different algorithm-independent learning schemes. Various versions of cross-validation and resampling techniques, leading to generation of a single classifier or a multiple classifier system, are scrutinised and compared. The classification performance on unseen data, commonly used as a criterion for comparing different competing designs, is augmented by further four criteria attempting to capture various additional characteristics of classifier generation schemes. These include: the ability to estimate the true classification error rate, the classifier transparency, the computational complexity of the learning scheme and the potential for adaptation to changing environments and new classes of data. One of the main questions examined is whether and when to use a single classifier or a combination of a number of component classifiers within a multiple classifier system.

© 2003 Elsevier B.V. All rights reserved.

*Keywords:* Neuro-fuzzy classifier; Pattern recognition; Ensembles of classifiers; Resampling techniques; Classifier combination; Cross-validation

---

## 1. Introduction

With an increasing computer power available at affordable prices and availability of vast amount of data there is an increasing need for robust methods and systems, which can take advantage of all available information. Automatic model building directly from data with a minimal or no human supervision is already absolutely crucial in order to stay competitive and maximally exploit the data

---

<sup>☆</sup> This paper is an invited extended version of a paper with the same title presented at the EUNITE'2001 Conference, Tenerife, Spain, 2001.

\* Tel.: +44-1202-595298; fax: +44-1202-595314.

*E-mail address:* [bgabrys@bournemouth.ac.uk](mailto:bgabrys@bournemouth.ac.uk) (B. Gabrys).

in quickly changing business environments. However, the methodology for ensuring that created models (i.e. classifiers, predictors, etc.) are as good as possible should be in place before using them with confidence.

No human supervision in model building also implies that one should use powerful enough techniques which can learn the data to any degree of accuracy. There are currently a lot of methods from soft computing, machine learning, and statistics domains which, in principal, satisfy this requirement. In the pattern recognition domain the examples include the nearest-neighbour classifiers [19,21], decision trees [19], neural networks with sufficient number of hidden nodes [4,14,19], fuzzy if-then rules systems which are built directly from data [1–3,16–18], neuro-fuzzy techniques based on hyperbox fuzzy sets [8–13], Bayesian networks or logical rule bases. From the statistical point of view most of these methods could be classified as non-parametric models. The main challenge in such cases is to design a model building strategy which would guard against overfitting of the training data or, in other words, would lead to a good generalisation performance.

Over the last 10 years, there has been a great amount of interest in the combination of the learning capability and computational efficiency of neural networks with the fuzzy sets ability to cope with uncertain or ambiguous data. This has led to a development of various hybrid neuro-fuzzy techniques [1–3,16–18], including a general fuzzy min-max (GFMM) neural network for clustering and classification [8–13]. The development of the GFMM originated from our investigation into uncertain information processing in the context of the decision support for the operational control of industrial processes. This generic pattern recognition method based on hyperbox fuzzy sets combines supervised and unsupervised learning within a single learning algorithm, can grow to meet the demands of the problem, has the ability to incorporate new information without a need for complete retraining, learns on-line and has the ability to process inputs in the form of real (confidence) intervals. It has been successfully applied to a very challenging problem of leakage detection and identification in water distribution systems where a hierarchical system based on the GFMM has been used [12].

Since proposing the original GFMM neural network a number of extensions have been proposed in an attempt to produce a flexible pattern recognition framework which could accommodate various problems when generating models directly from high-dimensional real-world data. These extensions included a development of agglomerative learning algorithms for GFMM resulting in a generation of hierarchical structures [11], various approaches to dealing with missing data [10] and the use of statistical resampling techniques in the process of generating classifiers with good generalisation performance through: data editing techniques [8], combining multiple copies of the GFMM classifier at the decision and model levels [9], and estimating the parameters controlling the complexity of the final GFMM classifier [11].

Though commonly the main objective of the classifier design process is constructing a classifier with as good a performance as possible, in many pattern classification applications there are some additional aspects which are regarded as equally important. For instance, it may be a part of the requirements that the classifier model is transparent and has the ability to provide an easily interpretable explanation of the suggested classification decision to a non-technical user. On the other hand, there may be applications where the speed of generation of the model is of primary concern or there are restrictions on the size of the classification model that can be stored. Yet another example could be an application where due to the non-stationary environment the emphasis should be put on the potential adaptability of the classifier model while in operation.

Bearing this in mind, in this paper we will concentrate on the analysis of various algorithm independent classifier model generation approaches for designing a GFMM classifier [8–13] or a GFMM-based multiple classifier system. Various model generation approaches, which could be used together with the base GFMM learning algorithms, will be assessed and discussed in the context of the following four criteria: (a) the ability to estimate the performance of the model on unseen data; (b) the generated model's power to explain the suggested decisions which can be interpreted by the user; (c) the computational complexity involved in the classifier model building process; and (d) the potential for adaptation of the classifier model to changing environments and new classes of data.

Specifically, the above criteria will be applied to five GFMM classifier model building schemes including generating classifiers on the basis of the full training data set, using a  $k$ -fold and multiple 2-fold cross-validation with various pruning approaches and combining multiple copies of the GFMM classifier.

Though in terms of pure classification performance the ensemble/combination methods [5–7,9,20] have been frequently shown to offer high classification performance gains in comparison to individual classifiers, when they are considered in the context of other criteria like the ones mentioned earlier the choice of the classifier generation scheme is no longer so clear. In this sense one of the main questions investigated will be that of whether to use a single model or a combination of a number of components forming the final classifier. As it will be illustrated each of the discussed model generation approaches has some advantages and disadvantages which make them more suitable for certain applications.

The remaining of this paper is organised as follows. The next section presents a summary of the GFMM neural network with definitions of hyperbox fuzzy sets and associated fuzzy membership function. It also provides a description of the base learning algorithms which can be used to place and adjust hyperboxes in the input space. In the following section different classifier generation schemes utilising statistical resampling techniques are discussed. This is followed by some simulation results illustrating their properties. And finally, conclusions are presented in the final section.

## 2. GFMM neural network

The GFMM neural network for classification constitutes a pattern recognition approach that is based on hyperbox fuzzy sets. A hyperbox defines a region of the  $n$ -dimensional pattern space, and all patterns contained within the hyperbox have full-class membership. A hyperbox is completely defined by its min- and max-points. The combination of the min–max points and the hyperbox membership function defines a fuzzy set. Learning in the GFMM neural network for classification consists of creating and adjusting hyperboxes in the pattern space. Once the network is trained the input space is covered with hyperbox fuzzy sets. Individual hyperboxes representing the same class are aggregated to form a single fuzzy set class. Hyperboxes belonging to the same class are allowed to overlap while hyperboxes belonging to different classes are not allowed to overlap therefore avoiding the ambiguity of an input having full membership in more than one class. The input to the GFMM can be itself a hyperbox (thus representing features given in a form of upper and lower limits) and is defined as follows:

$$X_h = [X_h^l \ X_h^u], \quad (1)$$

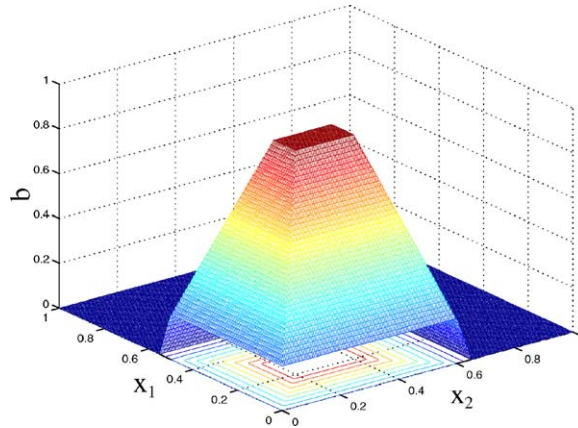


Fig. 1. A graphical illustration of the hyperbox membership function (3) for the values of  $V = [0.2 \ 0.2]$ ,  $W = [0.3 \ 0.4]$  and  $\gamma = 4$ .

where  $X_h^l$  and  $X_h^u$  are the lower and the upper limit vectors for the  $h$ th input pattern. Inputs are contained within the  $n$ -dimensional unit cube  $I^n$ . When  $X_h^l = X_h^u$  the input represents a point in the pattern space.

The  $j$ th hyperbox fuzzy set,  $B_j$  is defined as follows:

$$B_j = \{V_j, W_j, b_j(X_h, V_j, W_j)\} \tag{2}$$

for all  $j = 1, 2, \dots, m$ , where  $V_j = (v_{j1}, v_{j2}, \dots, v_{jn})$  is the min-point for the  $j$ th hyperbox,  $W_j = (w_{j1}, w_{j2}, \dots, w_{jn})$  is the max-point for the  $j$ th hyperbox, and the membership function for the  $j$ th hyperbox is

$$b_j(X_h, V_j, W_j) = \min_{i=1, \dots, n} (\min([1 - f(x_{hi}^u - w_{ji}, \gamma_i)], [1 - f(v_{ji} - x_{hi}^l, \gamma_i)])), \tag{3}$$

where

$$f(x, \gamma) = \begin{cases} 1 & \text{if } x\gamma > 1, \\ x\gamma & \text{if } 0 \leq x\gamma \leq 1, \\ 0 & \text{if } x\gamma < 0, \end{cases}$$

two parameter ramp threshold function,  $\gamma = [\gamma_1, \gamma_2, \dots, \gamma_n]$  is the sensitivity parameters governing how fast the membership values decrease; and  $0 \leq b_j(X_h, V_j, W_j) \leq 1$ . A graphical example of the membership function is shown in Fig. 1. Since the individual dimensions of a hyperbox fuzzy set are represented by trapezoidal membership functions each of the hyperbox fuzzy sets could be converted into a fuzzy rule and the GFMM into a fuzzy rule-based classifier [1,3,16–18].

The hyperbox membership values for each of the  $p$  classes are aggregated using the following formula:

$$c_k = \max_{j=1}^m b_j u_{jk}, \tag{4}$$

where  $U$  is the binary matrix with values  $u_{jk}$  equal to 1 if the  $j$ th hyperbox fuzzy set is a part of the  $k$ th class and 0 otherwise; and  $c_k \in [0, 1]$ ,  $k = 1, \dots, p$ , represent the degrees of membership of

the input pattern in the  $k$ th class. A single winning class can be found by finding the maximum value of  $c_k$ .

### 3. GFMM learning algorithms

Two principal learning approaches have been developed which can be used while training GFMM classifiers: an incremental learning [13] and an agglomerative learning [11].

The incremental learning can be described as a dynamic hyperbox expansion/contraction process where hyperbox fuzzy sets are created and adjusted in the pattern space after every presentation of an individual training pattern. A general strategy adopted is that of allowing to create relatively large clusters of data (i.e. hyperboxes) in the early stages of learning and reducing (if necessary) the maximum allowable size of the clusters (i.e. hyperboxes) in subsequent learning runs in order to accurately capture complex non-linear boundaries between different classes.

In contrast to the on-line version, the data clustering process using the agglomerative algorithm can be described as a bottom-up approach where one starts with very small clusters (i.e. individual data patterns) and builds larger representations of groups of original data (i.e. hyperboxes) by aggregating smaller clusters (i.e. hyperboxes/individual data patterns).

As it was explained in [11] these two types of learning algorithms have a number of complementary features with the incremental learning more suitable for on-line adaptation and dealing with large training data sets while agglomerative learning showing robust behaviour in presence of noise and outliers and insensitivity to the order of training patterns presentation.

Both types of learning algorithms share the ability to process labelled and unlabelled input data which is reflected in their definitions.

The input data used during the training stage of GFMM is specified as a set of  $N$  ordered pairs

$$\{\mathbf{X}_h, d_h\}, \quad (5)$$

where  $\mathbf{X}_h$  is the  $h$ th input pattern as described in (1) and  $d_h \in \{0, 1, 2, \dots, p\}$  is the index of one of the  $p + 1$  classes, where  $d_h = 0$  means that the input vector is unlabelled.

This forms the basis for the following learning algorithms.

#### 3.1. Incremental learning

The incremental learning algorithm is a four-step process consisting of *initialisation*, *expansion*, *overlap test*, and *contraction* with the last three steps repeated for each training input pattern. While the rational and detailed discussion of each of the four steps is included in [13], they are briefly described below.

##### 3.1.1. Initialisation

When a new hyperbox needs to be created its min,  $\mathbf{V}_j$ , and max,  $\mathbf{W}_j$ , points are initialised in such a way that the hyperbox adjusting process used in the expansion part of the learning algorithm can be automatically used. The  $\mathbf{V}_j$  and  $\mathbf{W}_j$  are set initially to

$$\mathbf{V}_j = \mathbf{1} \quad \text{and} \quad \mathbf{W}_j = \mathbf{0}. \quad (6)$$

This initialisation means that when the  $j$ th hyperbox is adjusted for the first time using the input pattern  $X_h = [X_h^l \ X_h^u]$  the min- and max-points of this hyperbox would be

$$V_j = X_h^l \quad \text{and} \quad W_j = X_h^u \tag{7}$$

identical to the input pattern.

### 3.1.2. Hyperbox expansion

When the  $h$ th input pattern  $X_h$  is presented, the hyperbox  $B_j$  with the highest degree of membership and allowing expansion (if needed) is found. The expansion criterion, that has to be met before the hyperbox  $B_j$  can expand to include the input  $X_h$ , consists of the following two parts:

(a) a test for the maximum allowable hyperbox size ( $0 \leq \Theta \leq 1$ ):

$$(\max(w_{ji}, x_{hi}^u) - \min(v_{ji}, x_{hi}^l)) \leq \Theta \quad \text{for all } i = 1, \dots, n \tag{8}$$

and

(b) a test for the class compatibility

if  $d_h = 0$  then *adjust*  $B_j$   
 else

$$\text{if } class(B_j) = \begin{cases} 0 \Rightarrow \text{adjust } B_j, \\ d_h \Rightarrow \text{adjust } B_j, \\ \text{else} \Rightarrow \text{take} \\ \text{another } B_j, \end{cases} \tag{9}$$

with the *adjust*  $B_j$  operation defined as:

$$v_{ji}^{new} = \min(v_{ji}^{old}, x_{hi}^l) \quad \text{for each } i = 1, \dots, n,$$

$$w_{ji}^{new} = \max(w_{ji}^{old}, x_{hi}^u) \quad \text{for each } i = 1, \dots, n.$$

If neither of the existing hyperboxes include or can expand to include the input  $X_h$ , then a new hyperbox  $B_k$  is created (see *initialization*), adjusted and labelled by setting  $class(B_k) = d_h$ .

### 3.1.3. Overlap test

Assuming that hyperbox  $B_j$  was expanded in the previous step, test for overlapping with  $B_k$  if

$$class(B_j) = \begin{cases} \text{test for overlapping} \\ 0 \Rightarrow \text{with all the other} \\ \text{hyperboxes,} \\ \text{else} \Rightarrow \text{test for overlapping} \\ \text{only if } class(B_j) \neq class(B_k). \end{cases} \tag{10}$$

### 3.1.4. Contraction

If an undesired overlap between two hyperboxes has been detected it is resolved by adjusting the two overlapping hyperboxes only along the dimension with the smallest overlap. Four possible cases for overlapping and contraction procedures are discussed in [13].

### 3.2. Agglomerative learning

The agglomerative learning for the GFMM neural network [11] initialises the min-point matrix  $V$  and the max-point matrix  $W$  to the values of the training set patterns lower  $X^l$  and upper  $X^u$  limits, respectively.

The hyperboxes are then agglomerated sequentially (one pair at a time) on the basis of the maximum similarity value calculated using the following similarity measure  $s_{jh} = s(\mathbf{B}_j, \mathbf{B}_h) = s_j(\mathbf{B}_h, V_j, W_j)$  between two hyperbox fuzzy sets  $B_h$  and  $B_j$ :

$$s_{jh} = \min_{i=1, \dots, n} (\min([1 - f(w_{hi} - w_{ji}, \gamma_i)], [1 - f(v_{ji} - v_{hi}, \gamma_i)])) \quad (11)$$

which has been adopted directly from (3) and takes into account not only the proximity of two hyperbox fuzzy sets but also their sizes. Two other similarity measures for hyperbox fuzzy sets are defined in [11] and although any of the similarity measures could be used the results presented in this paper have been obtained using (11). Since in general when using (11)  $s_{jh} \neq s_{hj}$ , i.e. a degree of similarity of  $B_h$  to  $B_j$  is not equal to a degree of similarity of  $B_j$  to  $B_h$  (with exception when  $B_h$  and  $B_j$  are points and some other special cases), the selection of a hyperbox  $B_l$ , to be aggregated with  $B_j$  is made by finding the maximum value from either: (a) the minimum similarity values  $\min(s_{jl}, s_{lj})$ ; or (b) the maximum similarity values  $\max(s_{jl}, s_{lj})$  among all possible pairs of hyperboxes ( $B_j, B_l$ ). In this paper we have used option (b) which leads to an agglomerative procedure somewhat resembling a single-link agglomerative algorithm [21]. The process of finding  $B_l$  can be summarised as follows:

$$s_{jh} = \max(\max(s_{jl}, s_{lj})) \quad \text{for all } l = 1, \dots, m, l \neq j. \quad (12)$$

The hyperboxes with the highest similarity value are only agglomerated if:

(a) newly formed hyperbox does not exceed the maximum allowable hyperbox size  $0 \leq \Theta \leq 1$

$$(\max(w_{ji}, w_{hi}) - \min(v_{ji}, v_{hi})) \leq \Theta \quad \text{for all } i = 1, \dots, n \quad (13)$$

and/or the similarity between hyperboxes  $B_h$  and  $B_j$  is greater than a certain user-defined similarity threshold value  $0 \leq s_{\min} \leq 1$

$$s_{jh} \geq s_{\min}, \quad (14)$$

(b) the agglomeration does not result in an overlap with any of the hyperboxes representing other classes (please see [13] for full details of the overlap test); and

(c) the hyperboxes  $B_h$  and  $B_j$  form a part of the same class or one or both are unlabelled which can be summarised as follows:

$$\begin{aligned} &\text{if } class(B_h) = 0 \text{ then aggregate } B_h \text{ and } B_j \\ &\text{else} \\ &\text{if } class(B_j) = \begin{cases} 0 \Rightarrow \text{aggregate } B_h \text{ and } B_j, \\ \text{class}(B_j) = \text{class}(B_h), \\ \text{class}(B_h) \Rightarrow \text{aggregate } B_h \text{ and } B_j, \\ \text{else} \Rightarrow \text{take another pair of hyperboxes.} \end{cases} \end{aligned} \quad (15)$$

If the above conditions are met the aggregation is carried out in the following way:

- (a) update  $B_j$  so that a new  $B_j$  will represent aggregated hyperboxes  $B_h$  and  $B_j$

$$v_{ji}^{\text{new}} = \min(v_{ji}^{\text{old}}, v_{hi}^{\text{old}}) \quad \text{for each } i = 1, \dots, n, \quad (16)$$

$$w_{ji}^{\text{new}} = \max(w_{ji}^{\text{old}}, w_{hi}^{\text{old}}) \quad \text{for each } i = 1, \dots, n, \quad (17)$$

- (b) remove  $B_h$  from a current set of hyperbox fuzzy sets.

The process of agglomeration is repeated until there are no hyperboxes which can be aggregated. The agglomeration of hyperboxes can be controlled by specifying different values for the maximum hyperbox size  $\Theta$  and hyperbox similarity threshold  $s_{\min}$  during the training process. For instance, in order to encourage creation of clusters in the densest areas first (i.e. aggregation of the most similar hyperboxes)  $s_{\min}$  can be initially set to a relatively high value and reduced in steps after all possible hyperboxes for a higher level of  $s_{\min}$  have been aggregated. In this way we are able to produce (simulate) a hierarchy of nested clusterings. The optimal value of  $s_{\min}$  when designing a GFMM classifier can be selected during a cross-validation procedure where a system with the smallest error for the validation set is sought.

For further details concerning the agglomerative learning and its potential use with the incremental version please refer to [11].

#### 4. Algorithm independent learning approaches

One of the most successful and commonly used approaches to estimating “true” errors and avoiding overfitting are based on various versions of cross-validation, bootstrap techniques and statistical resampling theory. While these resampling techniques have strong theoretical foundations they are also applicable to virtually any learning method.

The resampling techniques have not only been used for error estimation but also for model building and improving classification performance. The description of five different model building schemes with the agglomerative algorithm used as a base learning algorithm follows.

##### 4.1. Generating GFMM classifier model on the basis of a full training data set without pruning procedures

The simplest way to generate a GFMM classifier is to apply the agglomerative algorithm to the whole training set. After the training is completed the training set is learnt perfectly i.e. with the zero resubstitution error rate. The result is similar to the approach employed with the nearest-neighbour classifiers which use the training set as the reference set or the rule-based classifiers used in [15]. However, the resubstitution error rate is a very poor estimate of the error which one can expect when testing on unseen data. Another problem with this approach is that the training data is very likely to be overfitted and a poor generalisation performance will ensue. Some of the generated hyperboxes can be overspecialised and representing only noisy data or outliers as illustrated in Fig. 2a. This in turn can make the interpretation of the classification results more difficult.

The advantage of this approach is that the model generation is very quick and does not require any additional procedures for hyperbox pruning. Since the GFMM can grow to accommodate new data, the adaptation which would result in creating new and adjusting the existing hyperboxes can be carried out using the base learning algorithm.

#### 4.2. *K-fold cross-validation with pruning procedures [11]*

In order to avoid overfitting and provide a better estimation of the generalisation error various cross-validation procedures can be used.

The basic idea in a simple cross-validation (also known as a train-and-test or holdout method) is to randomly split the set of training samples into two parts: first which is used as the traditional training set for adjusting model parameters and second, the validation set, which is used to estimate the generalisation error. It is essential that the validation (or the testing) set does not include the points used for adjusting the model parameters—a methodological error known as “testing on the training set” which usually would lead to overoptimistic estimates of the generalisation error.

The single train-and-test method (or holdout method) is the technique which is easiest to analyse and the one with clear theoretical results. The test sample error rate is far stronger than the apparent error rate (obtained when testing on the training set). With large numbers of samples it is a very reasonable approach to model building and testing.

With more moderately sized samples, the holdout method usually leaves one with either insufficient training or testing cases. Not all the cases are used for model building and the method is subject to the idiosyncrasies of a single random train-and-test partition. Clearly, the test cases contain useful information for learning. Yet, they are ignored for training purposes.

While the single train-and-test method is the simplest technique for “honestly” estimating error rates, a single random partition can be misleading for small or moderately sized samples, and multiple train-and-test experiments can do better.

When multiple random train-and-test experiments are performed, a new model is learned from each training set. The estimated error rate is the average of the error rates for models derived for the independently and randomly generated test partitions. Random resampling solves the problem of relying on a single and possibly uncharacteristic partition by averaging the results over many randomly generated train-and-test partitions. The specific examples of resampling include the  $k$ -fold cross-validation error estimation method, in which the cases are randomly divided into  $k$  mutually exclusive set partitions of approximately equal size. The great advantage of  $k$ -fold cross-validation is that all the cases in the available sample are used for testing.

Apart from a better estimate of the true error the complexity of the GFMM classifier can be controlled by applying a pruning procedure. The basic pruning procedure used in this paper removes hyperbox fuzzy sets which cause more misclassifications than correct classifications on the validation set. In this way a simpler classifier can be generated with larger hyperboxes which can be more easily interpreted.

The disadvantage of this method is that not all the training data are used for the classifier design and that one cannot say which of the  $k$ -generated classifiers should be delivered as the final model. The model adaptation, as well as in the methods described below, can be problematic. On the one hand, if the new data were to be included in the model immediately when it is presented, the on-line learning algorithm could be applied but the classifier would quickly degenerate to the case where

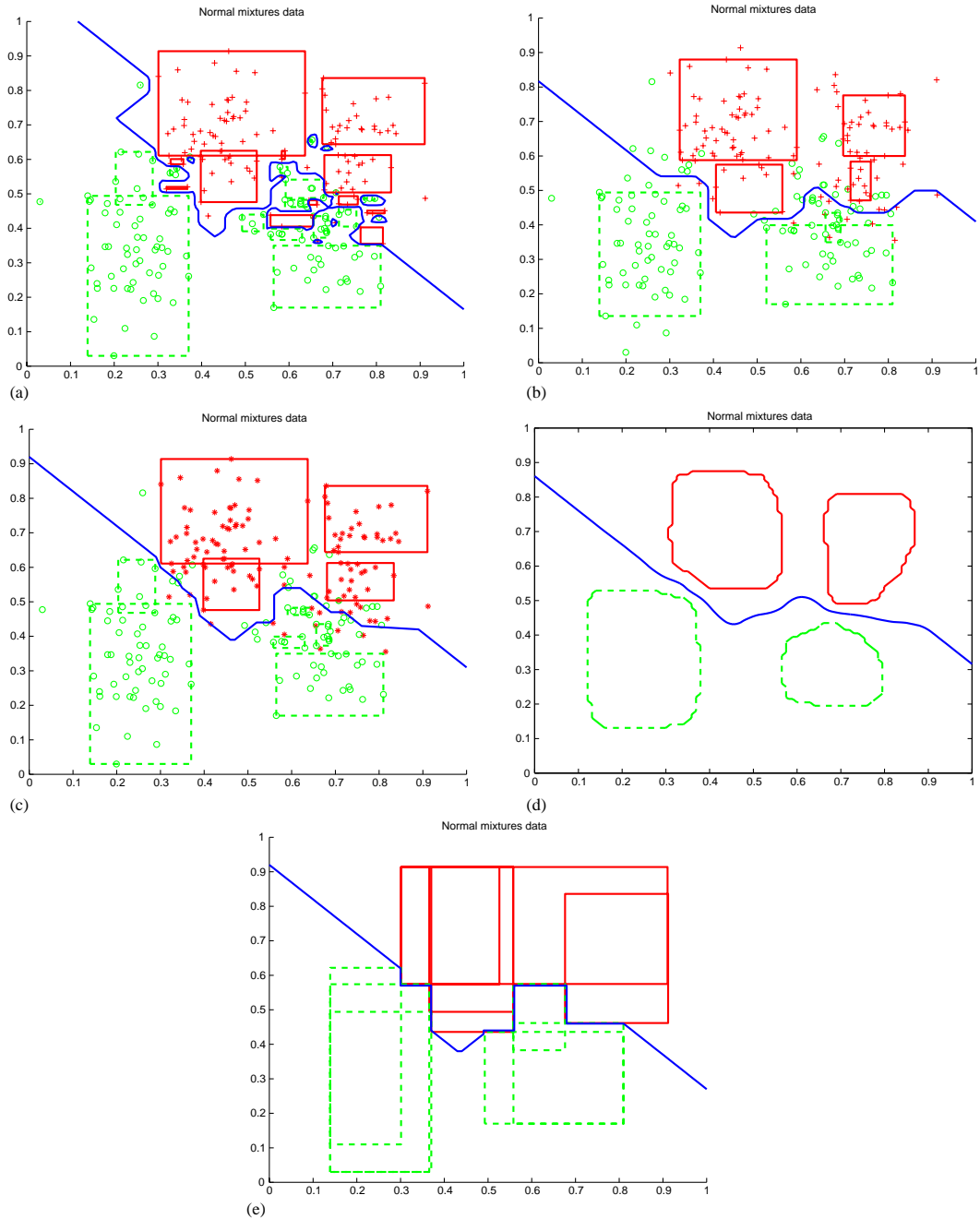


Fig. 2. The hyperboxes formed and the decision boundaries for the normal mixtures data set based on: (a) applying agglomerative learning to the full training data set; (b) single two-fold cross-validation procedure; (c) multiple two-fold cross-validation and hyperbox cardinality-based pruning; (d) combination at the decision level of 40 copies of GFMM generated during multiple two-fold cross-validation; (e) combination at the model level of 40 copies of GFMM generated during multiple two-fold cross-validation.

no pruning was used. On the other hand, an evolving validation set accounting for new data could be used for periodical pruning and model updating.

#### 4.3. *Multiple two-fold cross-validation with pruning procedures [9,11]*

While  $k$ -fold cross validation provides good, true error estimates for medium-sized problems, repeated two-fold cross-validation can provide yet more accurate true error estimates especially for small number of training samples. Additional advantage of using multiple two-fold cross-validation is the opportunity for a better estimation of some parameters for controlling the complexity of the final model or identifying noisy samples from the original training set which are most likely to be the cause of a poor generalisation performance.

One such approach discussed in [8] is based on a modified version of the data editing procedures which are commonly used with the  $k$ -nearest-neighbour classifiers. In case of  $k$ -nearest-neighbour classifiers the data editing procedures are usually applied with the aims of increasing the computational efficiency, through reduction of the number of reference data samples, and improving the generalisation performance through filtering out the outliers and noisy samples. In [8] the training data editing procedure is based on estimating the probability of every single sample in the original training set to be used in the generation of the hyperboxes during the multiple cross-validation. This probability is simply a ratio of the number of times a training sample  $X_h$  has been used in generation of a hyperbox which is retained in the classifier model after the pruning to the total number of repetitions of the two-fold cross-validation. The training samples with small probability values are removed from the training set and the base learning algorithm is applied to the remaining training samples.

In this paper the repeated two-fold cross-validation is used for estimating the minimum cardinality (number of samples) of a hyperbox fuzzy set for which the hyperbox fuzzy set should be still retained in the final GFMM classifier model. It is based on the assumption that hyperboxes representing small number of data samples are most likely to cover noisy samples or outliers. Once the minimum cardinality is estimated the training is performed for the whole training set and hyperbox fuzzy sets representing a number of input patterns smaller than this minimum cardinality are pruned.

The model generated in this way is of the similar complexity as in the case of a single  $k$ -fold cross-validation but all training data are used in the training process. The transparency of the model is retained but the adaptation to new data could be more difficult if the whole process of multiple cross-validation was to be repeated on a regular basis.

#### 4.4. *Ensemble of classifiers obtained from repeated two-fold splitting of the training data set and averaging the outputs of individual classifiers [9]*

Even with a model complexity control in place, the variance component of the classification error for such highly flexible classifiers as the GFMM can be high. In recent years the studies of resampling techniques have led not only to developing techniques for a better estimation of the true classification error but also to the generation of multiple classifier systems known as ensemble classifiers.

Among the ensemble generation techniques based on resampling methods the most popular and widely used are bagging and boosting [5,7]. Both techniques rely on selective resampling of the training set in order to generate multiple copies of the classifier by repeatedly applying the base learning algorithm to different subsets of the original training set. The classifiers generated in this way

are then combined by either averaging or voting their decisions. It has been frequently illustrated that bagging, boosting and their variants are very effective in improving the performance and reliability of a derived classifier ensemble in comparison to the individual components.

In order to take advantage of the reduction of the variance and stabilising effect of bagging, the outputs of a multiple copies of the GFMM classifier generated during repeated two-fold cross-validation can be averaged in the following way:

$$c_k^* = \frac{1}{L} \sum_{i=1}^L c_{ki}, \quad (18)$$

where  $L$  is the number of classifiers in the ensemble and  $c_k^*$  is the average  $k$ th class membership value for  $k = 1, \dots, p$ .

However, the improved classification performance of an ensemble classifier comes at a cost of vastly increased complexity of the classification system and often increased time of achieving the classification results. The transparency of the classification decisions is also lost. The adaptation of the model to changing environments can be even more difficult since a number of copies of the GFMM classifier would have to be adapted at the same time.

#### 4.5. Combination of individual models obtained from repeated two-fold splitting of the training data set [9]

An alternative to combining at the decision level (i.e. combining outputs of multiple copies of the GFMM classifier) is a combination at the model level (i.e. combining the hyperbox fuzzy sets from different copies of the GFMM) introduced in [9].

The basic idea is to use the hyperbox fuzzy sets from all models to be combined as inputs to the base training algorithm. The possibility of reducing the complexity of the final model while preserving the stability and improved performance of the ensemble is based on the observation that many of the hyperbox fuzzy sets from different component classifiers would be redundant and therefore can be agglomerated since they cover the “trivial” areas of the input space while the subtly differing (complementary) hyperboxes covering the areas near the class boundaries or overlapping regions can be added and refined.

In this way the resulting GFMM classifier complexity and transparency is usually comparable with classifiers generated during a single cross-validation procedure while the improved classification performance and reduced variance is comparable to the ensemble of classifiers with combined decisions. This, however, comes at a cost of increased training time since additional training cycle is added. The adaptation of the model could be carried out as for any other single GFMM model but the benefits of the combination of the models could be quickly lost.

## 5. Experimental results

The above analysis of different GFMM classifier model generation schemes will now be illustrated on the basis of four non-trivial data sets representing different pattern classification problems.

The first two two-dimensional synthetic data sets represent cases of non-linear classification problems with highly overlapping classes and a number of data points which can be classified as outliers

Table 1  
The sizes of data sets used in the experiments

Data set	No. of inputs	No. of classes	No. of data points		
			Total	Train	Test
Normal mixtures	2	2	1250	250	1000
Cone-torus	2	3	800	400	400
IRIS	4	3	150	135	15
Wine	13	3	178	160	18

The IRIS and Wine data sets tested within a 10-fold cross-validation scheme.

Table 2  
Testing error rates (%) for four well-known classifiers and the examined data sets

Data set	Classifier			
	Quadratic discriminant classifier	Parzen	Nearest neighbour	Multilayer perceptron with backpropagation
Normal mixtures	10.2	10.9	15.0	9.2
Cone-torus	16.75	12.25	15.25	13.75
IRIS	2.61	4.09	4.0	4.39
Wine	3.24	3.41	3.41	3.16

or noisy samples. Using two-dimensional problems also offer a chance of visually examining the created class boundaries and illustrating the problems of data overfitting and model transparency. In addition these data sets have been used in a number of studies with tests carried out for a large number of different classifiers and multiple classifier systems [16,19].

The other two data sets have been obtained from the repository of machine learning databases (<http://www.ics.uci.edu/~mllearn/MLRepository.html>) and concern the problems of classifying iris plants (IRIS data set) and three types of wine (Wine data set).

The sizes and splits for training and testing for all five data sets are shown in Table 1. For the reference purposes some testing results for four well-known classifiers available in PRTOOLS 3.1 (<ftp://ftp.ph.tn.tudelft.nl/pub/bob/prtools>) are also shown in Table 2.

The first two-dimensional problem was introduced by Ripley [19]. The training data, shown in Fig. 2, consists of two classes with 125 points in each class. Each of the two classes has bimodal distribution and the classes were chosen in such a way as to allow the best-possible error rate of about 8%. The testing has been carried out on an independent testing set of 1000 samples drawn from the same distribution.

The second two-dimensional data set, shown in Fig. 3, has been introduced by Kuncheva [16] and used throughout the text of her book to illustrate the performance of various classification techniques. The cone-torus training data set consists of three classes with 400 data points generated from three differently shaped distributions: a cone, half a torus, and a normal distribution. The prior probabilities for the three classes are 0.25, 0.25 and 0.5. The training data and a separate

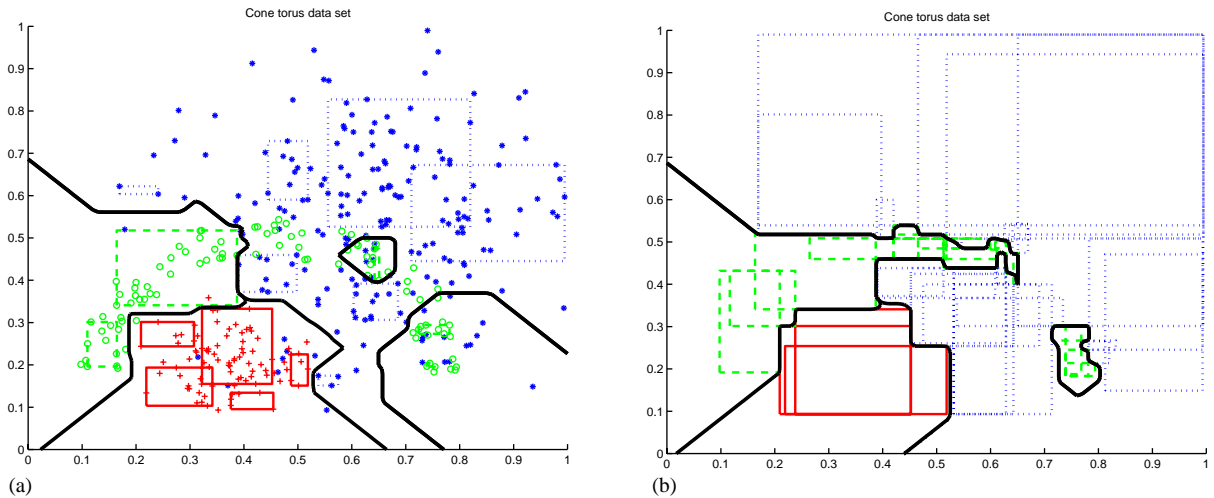


Fig. 3. Cone-torus data classification using GFMM classifier: (a) The training data, decision boundary and hyperboxes created during a single two-fold cross-validation; (b) the decision boundary and hyperboxes resulting from combining 40 copies of the GFMM classifier at the model level.

testing set consisting of further 400 samples drawn from the same distribution are available at <http://www.bangor.ac.uk/~mas00a/>.

The graphical illustrations of the generated models (hyperbox fuzzy sets) and obtained decision boundaries for the five options of generating individual and ensembles of GFMM classifiers for the normal mixtures data set are shown in Fig. 2. Examples of the hyperbox fuzzy sets generated during a single two-fold cross-validation and after combination of 40 copies of the GFMM classifier at the model level for the cone-torus data set are shown in Fig. 3.

As we can see from Figs. 2 and 3, each of the hyperboxes covers a part of the input space which can represent a specific class of data. The hyperbox min–max points can be easily converted into a set of rules understandable for non-technical user. The classification boundaries created can be of arbitrary shape. The hyperboxes can be quite easily expanded and shrunk to accommodate new data.

The simulation results for each of the five model generation schemes are shown in Tables 3, 4, 5 and 6 for the normal mixtures, cone-torus, IRIS and Wine data sets, respectively. As we can see from the tables the approaches which are based on multiple cross-validation offer significant improvement in the performance in comparison to the other model generation schemes. It is especially evident in the case of normal mixtures (Table 3) and Wine (Table 6) data sets. The performance improvement of the ensemble of classifiers (i.e. combination at the decision level) comes at a cost of vastly increased model complexity (i.e. number of hyperboxes in the final model) which dramatically increases with the increase of the component models combined. The model complexity of the classifier generated on the basis of the full training set without pruning is also significantly higher in comparison to the other schemes generating a single GFMM model. This is due to overfitting of the training data which is reflected in the classification performance and illustrated by too complex decision boundary for the normal mixtures data set shown in Fig. 2a. On the other hand, the hyperbox cardinality-based pruning when applied to the same model generated on the

Table 3  
GFMM classification results for the normal mixtures data set

Classifier generation procedure	No. of combined classifiers	Average No. of hyperboxes in the final model	Training set error rate (%)		Testing set error rate (%)	
			Mean error	Standard deviation	Mean error	Standard deviation
No validation, training on a full data set	1	37	0	—	12.1	—
Two-fold cross-validation	1	6.78	13.48	1.54	9.64	1.03
Multiple two-fold cross-validation + cardinality-based pruning	1	10	11.6	—	8.2	—
Combination at the decision level	40	270.25	13.2	0.57	8.55	0.25
Combination at the model level	40	15.75	9.8	1.48	8.15	0.24

Table 4  
GFMM classification results for the cone-torus data set

Classifier generation procedure	No. of combined classifiers	Average No. of hyperboxes in the final model	Training set error rate (%)		Testing set error rate (%)	
			Mean error	Standard deviation	Mean error	Standard deviation
No validation, training on a full data set	1	55	0	—	15	—
Two-fold cross-validation	1	17.11	12.54	1.69	14.78	1.57
Multiple two-fold cross-validation + cardinality-based pruning	1	18	11.00	—	11.00	—
Combination at the decision level	40	684.4	12.19	0.97	13.53	0.99
Combination at the model level	40	42.23	6.75	0.35	13.06	0.95

basis of the full training set can significantly improve the performance while reducing the model complexity.

An interesting case can be observed for the Wine data set (Table 6) where an increase in the number of hyperboxes when combining at the model level is associated with the decrease in the classification error. It is evident that in this case the additional hyperbox fuzzy sets do not contribute to the overfitting of the training data but exploit various combinations of the training samples in the areas around the class boundaries or overlapping regions.

In terms of transparency and interpretability the models generated using single two-fold cross-validation and cardinality-based pruning generally result in the smallest number of hyperboxes. The number of hyperboxes generated while combining at the model level is driven by the improvement in accuracy which for relatively simple problems like IRIS data set (Table 5) meant that the final number of hyperboxes could be even significantly smaller than for models generated during a single two-fold cross-validation.

Table 5  
GFMM classification results for the IRIS data set

Classifier generation procedure	No. of combined classifiers	Average No. of hyperboxes in the final model	Training set error rate (%)		Testing set error rate (%)	
			Mean error	Standard deviation	Mean error	Standard deviation
No validation, training on a full data set	1	22	0	—	4.67	0.21
Two-fold cross-validation	1	16.1	2.21	0.05	3.60	0.21
Multiple two-fold cross-validation + cardinality-based pruning	1	4	2.67	0.08	4.00	0.3
Combination at the decision level	40	639.9	0.53	0.2	3.87	0.3
Combination at the model level	40	5.6	1.2	0.8	3.33	0

Table 6  
GFMM classification results for the Wine data set

Classifier generation procedure	No. of combined classifiers	Average No. of hyperboxes in the final model	Training set error rate (%)		Testing set error rate (%)	
			Mean error	Standard deviation	Mean error	Standard deviation
No validation, training on the full data set	1	16	0	—	9.38	1.07
Two-fold cross-validation	1	9.82	6.11	0.34	8.28	0.65
Multiple two-fold cross-validation + cardinality-based pruning	1	7	4.44	0.3	5.68	0.71
Combination at the decision level	40	392.1	0.48	0.27	4.63	0.56
Combination at the model level	40	26	0.29	0.21	3.75	1.33

Model adaptation in each of the considered cases could be carried out using an incremental learning algorithm described in Section 3.1. However, that would mean that if the classifier would be allowed to run in an on-line learning mode for too-long, the benefits of the resampling techniques and the cross-validation schemes would be lost. An alternative could be a hybrid approach which would combine an incremental adaptation of the model with a periodical pruning based on an evolving training set. Such approaches are currently under investigation.

## 6. Conclusions

Five different GFMM classifier model generation approaches have been presented and discussed. Depending on a classification problem each of the approaches can be shown to have some advantages over another. If the speed of model generation and quick adaptability to the changing environment are of primary concern then the base learning algorithm can be used without any hyperbox pruning

procedures. On the other hand, if the model building can be carried out off-line, the approaches based on multiple cross-validation either for estimating parameters controlling the complexity of the model or for building an ensemble of classifiers are likely to provide a better classification performance and the true error estimation. If the explanation of the classification decisions is to be provided, all the model generation schemes resulting in a single GFMM model can be used. This is with the exception of the ensemble of GFMM classifiers in which case the single GFMM model transparency is lost. While currently the advantages of the resampling techniques can be fully realised only during the initial model building process, an evolving validation set accounting for new data could be used for periodical model updating and hyperbox pruning. The frequency of such model validation would be dependent on the dynamics of a specific pattern classification problem.

### Acknowledgements

Research reported in this paper has been supported by the Nuffield Foundation Grant (NAL/00259/G).

### References

- [1] S. Abe, M. Lang, A method for fuzzy rules extraction directly from numerical data and its application to pattern classification, *IEEE Trans. Fuzzy Systems* 3 (1) (1995) 18–28.
- [2] M. Berthold, Fuzzy models and potential outliers, *Proc. NAFIPS-99*, IEEE Press, New York, 1999, pp. 532–535.
- [3] J. Bezdek, J. Keller, R. Krisnapuram, N.R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer Academic Publishers, Dordrecht, 1999.
- [4] C.M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.
- [5] L. Breiman, Bagging predictors, *Mach. Learn.* 24 (2) (1996) 123–140.
- [6] T.G. Dietterich, Machine learning research: four current directions, *AI Mag.* 18 (4) (1997) 97–136.
- [7] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, *Mach. Learn.* 40 (2000) 139–157.
- [8] B. Gabrys, Data editing for neuro-fuzzy classifiers, *Proc. SOCO'2001 Conf.*, Abstract page 77, Paper no. #1824-036, Paisley, Scotland, June 2001, ISBN: 3-906454-27-4.
- [9] B. Gabrys, Combining neuro-fuzzy classifiers for improved generalisation and reliability, *Proc. Internat. Joint Conf. Neural Networks (IJCNN'2002) a part of the WCCI'2002 Congr.*, Honolulu, USA, May 2002, pp. 2410–2415, ISBN: 0-7803-7278-6.
- [10] B. Gabrys, Neuro-fuzzy approach to processing inputs with missing values in pattern recognition problems, *Internat. J. Approx. Reason.* 30 (3) (2002) 149–179.
- [11] B. Gabrys, Agglomerative learning algorithms for general fuzzy min–max neural network, *J. VLSI Signal Process. Systems* 32 (1–2) (2002) 67–82.
- [12] B. Gabrys, A. Bargiela, Neural networks based decision support in presence of uncertainties, *J. Water Resources Plann. Manage.* 125 (5) (1999) 272–280.
- [13] B. Gabrys, A. Bargiela, General fuzzy min–max neural network for clustering and classification, *IEEE Trans. Neural Networks* 11 (3) (2000) 769–783.
- [14] M.H. Hassoun, *Fundamentals of Artificial Neural Networks*, The MIT Press, Cambridge, MA, 1995.
- [15] K.-P. Huber, M. Berthold, Building precise classifiers with automatic rule extraction, *Proc. IEEE Internat. Conf. Neural Networks (IJCNN)*, Vol. 3, Perth, Australia, 1995, pp. 1263–1268.
- [16] L.I. Kuncheva, *Fuzzy Classifier Design*, Physica-Verlag, Heidelberg, 2000.
- [17] D. Nauck, R. Kruse, A neuro-fuzzy method to learn fuzzy classification rules from data, *Fuzzy Sets and Systems* 89 (3) (1997) 277–288.

- [18] D. Nauck, R. Kruse, Learning in neuro-fuzzy systems with symbolic attributes and missing values, Proc. Internat. Conf. Neural Information Processing—ICONIP'99, Perth, 1999, pp. 142–147.
- [19] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, 1996.
- [20] D. Ruta, B. Gabrys, An overview of classifier fusion methods, in: Prof. M. Crowe (Ed.), Computing and Information Systems, University of Paisley, Vol. 7(1), 2000, pp. 1–10.
- [21] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press, New York, 1999.