Journal of VLSI Signal Processing 32, 67–82, 2002

Agglomerative Learning Algorithms for General Fuzzy Min-Max Neural Network

BOGDAN GABRYS
Applied Computational Intelligence Research Unit, Division of Computing and Information Systems, University of Paisley, High Street, Paisley PA1 2BE, Scotland, UK

Received April 30, 2001; Revised September 19, 2001; Accepted November 19, 2001

Abstract. In this paper two agglomerative learning algorithms based on new similarity measures defined for hyperbox fuzzy sets are proposed. They are presented in a context of clustering and classification problems tackled using a general fuzzy min-max (GFMM) neural network. The proposed agglomerative schemes have shown robust behaviour in presence of noise and outliers and insensitivity to the order of training patterns presentation. The emphasis is also put on the complimentary features to the previously presented incremental learning scheme more suitable for on-line adaptation and dealing with large training data sets. The performance and other properties of the agglomerative schemes are illustrated using a number of artificial and real-world data sets.

Keywords: pattern classification, hierarchical clustering, agglomerative learning, neuro-fuzzy system, hyperbox fuzzy sets

1. Introduction

One of the most significant features of adaptive intelligent systems is their ability to learn. This learning is usually accomplished through an adaptive procedure, known as learning rule or algorithm, which gives a formula of updating the parameters of the system (i.e. adapting weights in artificial neural networks) in such a way as to improve some performance measure [1].

All learning algorithms to be found in the neural network and machine learning literature can be classified as supervised, unsupervised or reinforcement learning. The distinctive feature in this classification is a type and presence of a target signal associated with each input/training pattern received from the environment.

There are many different learning rules falling within each of these three categories. They in turn can be further divided into incremental (also known as sequential) or batch learning. In incremental learning the parameters are updated after every presentation of an input pattern. In the batch learning, on the other hand, the parameter updating is performed only after all training patterns have been taken into consideration. Each of these two approaches has some advantages over the other. Incremental learning is often preferred because: first, it requires less storage than batch learning which can prove very useful when large volumes of data are available; second, it can be used for 'on-line' learning in real-time adaptive systems; third, because of its stochastic character it can potentially escape from local minima and arrive at better-quality solutions. On the other hand, solutions found using incremental learning rules depend, to a greater or lesser extent, on the order of presentation of the training patterns, which in turn implies the sensitivity of such algorithms to initialization and greater sensitivity to noise and outliers. Stability and convergence of the incremental learning rules cannot be always proven which in extreme cases can mean divergence and a complete breakdown in the algorithm.

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 [2–6]. An example of such a combination is a general fuzzy min-max (GFMM) neural network for clustering and classification [7–10].

The GFMM NN for clustering and classification constitutes a pattern recognition approach that is based on hyperbox fuzzy sets [4, 5, 7–10]. The incremental learning proposed in [10] combines the supervised (classification) and unsupervised (clustering) learning within a single training algorithm. The training 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 in the early stages of learning and reducing (if necessary) the maximum allowable size of the clusters in subsequent learning runs in order to accurately capture complex nonlinear boundaries between different classes.

This strategy have been shown to work very well in most of the cases. However, it has also been found that the resulting input-output mapping depends on the order of presentation of the training patterns and the method is sensitive to noise and outliers.

Overlapping hyperboxes represent another undesired effect resulting from the dynamic nature of the incremental learning algorithm. Because hyperbox overlap causes ambiguity and creates possibility of one pattern fully belonging to two or more different classes, the overlaps have to be resolved through a contraction process. This effect occurs purely because hyperbox expansion decisions have to be made on the basis of a single (current) input pattern and quite often would not have been taken in the first place, had the whole training data been available at the same time.

In this paper two agglomerative learning algorithms for the GFMM neural network are proposed.

The agglomerative algorithms are part of a larger group of hierarchical clustering algorithms. Due to their philosophy of producing hierarchies of nested clusterings, they have been popular in a wide range of disciplines from biology, medicine and archeology to computer science and engineering [11–14]. From our point of view, the main advantages of hierarchical clustering procedures are their insensitivity to the order of data presentation and initialization, their graphical representation of clusterings in form of dendrograms which can be easily interpreted even for high dimensional data and their potential resistance to noise and outliers that will be exploited and illustrated in the later sections of this paper.

Taking into account the deficiencies observed in the previously presented incremental learning algorithm and the general strong qualities of hierarchical clustering procedures, the agglomerative learning algorithms for GFMM have been developed. These algorithms can be used as an alternative or compliment to the incremental version for an off-line training performed on finite training data sets. The mechanisms for processing labelled and unlabelled input data introduced for the incremental version are transferred into the agglomerative schemes ensuring that the hybrid clustering/classification character of GFMM is preserved.

In contrast to the incremental version, the data clustering process using the agglomerative algorithms 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 by aggregating smaller clusters. In this sense it can also be viewed as a neural network structure optimization method where an aggregation of two clusters means decreasing the number of neurons required to encode the data.

Most agglomerative algorithms to be found in the literature are based on point representatives of clusters and similarity measures defined for points [11, 14]. Some other similarity measures have been used with cluster representatives in a form of hyperspheres, hyperelipsoids or hyperplanes [12, 13]. In the agglomerative procedures proposed here, the similarity measures defined for hyperboxes used as cluster representatives are utilised. The first of the agglomerative schemes is based on the operations on a full similarity matrix though using one of the proposed similarity measures between hyperboxes generally results in asymmetric similarity matrices. It is in contrast to symmetric matrices normally encountered in other agglomerative algorithms (with exception of [15]). The potential implications and properties of the algorithm stemming from this fact will be discussed in Section 3. The second agglomerative scheme has been developed in response to high computational complexity associated with operations on and updating of the full similarity matrix. As it is well known from the literature [14], similarity (dissimilarity) matrix based agglomerative algorithms are of the $O(n^3)$ complexity which for large training data sets implies long training times. The improved training time of the second algorithm has been achieved by not using the full similarity matrix during the hyperbox

aggregation. This improvement came at a cost of the outcome of the second algorithm being dependant to some extent on the order of training data presentation though to a much lesser degree than the on-line version proposed previously.

The remaining of this paper is organised as follows. Section 2 provides a general description of GFMM neural network operation and definitions of hyperbox fuzzy sets used as cluster prototypes. In Section 3 the similarity measures for hyperbox fuzzy sets and the two agglomerative learning algorithms for GFMM are presented. Section 4 illustrates the performance and properties of the agglomerative learning for some artificial and real data sets used in pattern recognition problems. Finally, conclusions are presented in the last section.

2. GFMM Description

GFMM neural network for clustering and classification [10] is a generalisation of and extension to the fuzzy min-max neural networks developed by Simpson [4, 5]. The main changes in GFMM constitute the combination of unsupervised and supervised learning, associated with problems of data clustering and classification respectively, within a single learning algorithm and extension of the input data from a single point in n-dimensional pattern space to input patterns given as lower and upper limits for each dimension—hyperbox in n-dimensional pattern space.

The GFMM is represented by a three layer feedforward neural network shown in Fig. 1. It consists of $2n$ input layer nodes, m second layer nodes representing hyperbox fuzzy sets and $p + 1$ output layer nodes representing classes.

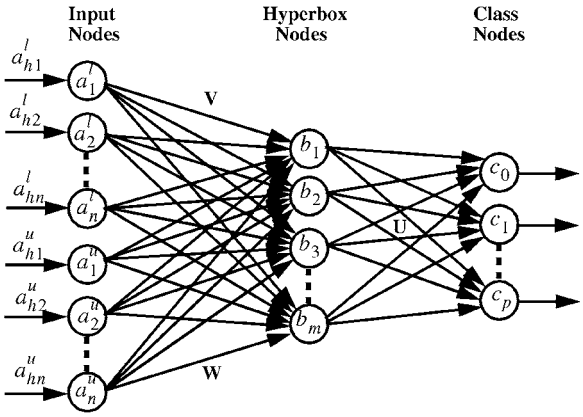

Figure 1. GFMM neural network for clustering and classification.

The basic idea of fuzzy min-max neural networks is to represent groups of input patterns using hyperbox fuzzy sets. A hyperbox fuzzy set is a combination of a hyperbox covering a part of n-dimensional pattern space and associated with it membership function. A hyperbox is completely defined by its min point and its max point. A membership function acts as a distance measure with input patterns having a full membership if they are fully contained within a hyperbox and the degree of membership decreasing with the increase of distance from the hyperbox. 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 following are the definitions of input data format, hyperbox fuzzy sets, hyperbox membership function and hyperbox aggregation formula that are used within GFMM.

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

$$\{\boldsymbol{A}_h, d_h\} \tag{1}$$

where $\boldsymbol{A}_h = [\boldsymbol{A}_h^l \ \boldsymbol{A}_h^u]$ is the h-th input pattern in a form of lower, $\boldsymbol{A}_h^l = (a_{h1}^l, a_{h2}^l, \ldots, a_{hn}^l)$, and upper, $\boldsymbol{A}_h^u = (a_{h1}^u, a_{h2}^u, \ldots, a_{hn}^u)$, limits vectors contained within the n-dimensional unit cube I^n; and $d_h \in \{0, 1, 2, \ldots, p\}$ is the index of one of the $p + 1$ classes, where $d_h = 0$ means that the input vector is unlabelled.

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

$$B_j = \{\boldsymbol{V}_j, \boldsymbol{W}_j, b_j(\boldsymbol{A}_h, \boldsymbol{V}_j, \boldsymbol{W}_j)\} \tag{2}$$

for all $j = 1, 2, \ldots, m$, where $\boldsymbol{V}_j = (v_{j1}, v_{j2}, \ldots, v_{jn})$ is the min point for the j-th hyperbox, $\boldsymbol{W}_j = (w_{j1}, w_{j2}, \ldots, w_{jn})$ is the max point for the j-th hyperbox, and the membership function for the j-th hyperbox is:

$$b_j(\boldsymbol{A}_h, \boldsymbol{V}_j, \boldsymbol{W}_j) = \min_{i=1..n} \big(\min\big(\big[1 - f\big(a_{hi}^u - w_{ji}, \gamma_i\big)\big], \big[1 - f\big(v_{ji} - a_{hi}^l, \gamma_i\big)\big]\big)\big) \tag{3}$$

where:

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

is two parameter ramp threshold function; $\gamma = [\gamma_1, \gamma_2, \ldots, \gamma_n]$ are sensitivity parameters governing how fast the membership values decrease; and $0 \leq b_j(\boldsymbol{A}_h, \boldsymbol{V}_j, \boldsymbol{W}_j) \leq 1$. For the simplicity we will refer to $b_j(\boldsymbol{A}_h, \boldsymbol{V}_j, \boldsymbol{W}_j)$ as b_j in the remaining of the paper.

Hyperbox fuzzy sets from the second layer are aggregated using the aggregation formula (4) in order to generate an output which represents the degree to which the input pattern $\boldsymbol{A}_h$ fits within the class k. The transfer function for each of the third layer nodes is defined as

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

for each of the $p+1$ third layer nodes. Node c_0 represents all unlabelled hyperboxes from the second layer. Matrix $\boldsymbol{U}$ represents connections between the hyperbox and class layers of the network and the values of $\boldsymbol{U}$ are assigned as follows:

$$u_{jk} = \begin{cases} 1 & \text{if } B_j \text{ is a hyperbox for class } c_k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

2.1. *Summary of the On-Line Learning Algorithm*

Since the on-line learning algorithm will be used together with the proposed agglomerative schemes, its brief summary follows.

The on-line learning for the GFMM neural network consists of creating and expanding/contracting hyperboxes in a pattern space. The learning process begins by selecting an input pattern and finding the closest hyperbox to that pattern that can expand (if necessary) to include the pattern. If a hyperbox cannot be found that meets the expansion criteria, a new hyperbox is formed and added to the system. This growth process allows existing clusters/classes to be refined over time, and it allows new clusters/classes to be added without retraining. One of the undesirable effects of hyperbox expansion are overlapping hyperboxes. Because hyperbox overlap causes ambiguity and creates possibility of one pattern fully belonging to two or more different clusters/classes, a contraction process is utilized to eliminate any undesired hyperbox overlaps.

In summary, the on-line learning algorithm is a four-step process consisting of *Initialization, 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 [10], they are briefly described below.

2.1.1. Initialization. When a new hyperbox needs to be created its min, $\boldsymbol{V}_j$, and max, $\boldsymbol{W}_j$, 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 $\boldsymbol{V}_j$ and $\boldsymbol{W}_j$ are set initially to:

$$\boldsymbol{V}_j = \boldsymbol{1} \quad \text{and} \quad \boldsymbol{W}_j = \boldsymbol{0} \tag{6}$$

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

$$\boldsymbol{V}_j = \boldsymbol{A}_h^l \quad \text{and} \quad \boldsymbol{W}_j = \boldsymbol{A}_h^u \tag{7}$$

identical to the input pattern.

2.1.2. Hyperbox Expansion. When the h-th input pattern $\boldsymbol{A}_h$ 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 $\boldsymbol{A}_h$, consists of the following two parts:

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

$$\begin{aligned} &\left(\max\left(w_{ji}, a_{hi}^u\right) - \min\left(v_{ji}, a_{hi}^l\right)\right) \leq \Theta \\ &\quad \text{for all } i = 1 \ldots n \end{aligned} \tag{8}$$

and

(b) a test for the class compatibility

if $d_h = 0$ then adjust B_j

else

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

with the adjust B_j operation defined as:

$$\begin{aligned} v_{ji}^{new} &= \min\left(v_{ji}^{old}, a_{hi}^l\right) \quad \text{for each } i = 1, \ldots, n \\ w_{ji}^{new} &= \max\left(w_{ji}^{old}, a_{hi}^u\right) \quad \text{for each } i = 1, \ldots, n \end{aligned}$$

If neither of the existing hyperboxes include or can expand to include the input $\boldsymbol{A}_h$, then a new hyperbox B_k is created (see *Initialization*), adjusted and labelled by setting $\text{class}(B_k) = d_h$.

2.1.3. Overlap Test. Assuming that hyperbox B_j was expanded in the previous step, test for overlapping with B_k if

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

2.1.4. 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 [10].

3. New Learning Algorithms

3.1. Similarity Measures

The membership function (3) has been designed to account for maximum violation of hyperbox min and max points by an input pattern $\boldsymbol{A}_h$. It has been dictated by the engineering application [9] where the worst possible case needed to be considered when classifying inputs in a form of lower and upper limits. While (3) is interpreted as a degree of belonging of $\boldsymbol{A}_h$ in a hyperbox fuzzy set B_j it can be easily adapted as a measure of similarity between two hyperbox fuzzy sets B_h and B_j.

For the agglomerative learning algorithms described next the following three similarity measures between two hyperboxes derived from (3) are proposed:

1. The first similarity measure between hyperboxes B_h and B_j, $s_{jh} = s(B_j, B_h)$ is taken directly from (3) and takes the following form:

$$s(B_j, B_h) = \min_{i=1..n} (\min([1 - f(w_{hi} - w_{ji}, \gamma_i)], [1 - f(v_{ji} - v_{hi}, \gamma_i)])) \tag{11}$$

   The characteristic features of this similarity measure are:

   (a) $s_{jj} = 1$
   (b) $0 \leq s_{jh} \leq 1 - s_{jh} = 1$ only if B_h is completely contained within B_j
   (c) $s_{jh} \neq s_{hj}$—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).

   The properties (c) and (a) lead to an asymmetrical similarity matrix used in the agglomerative algorithms with ones on its diagonal.

2. The second similarity measure between hyperboxes B_h and B_j, $\tilde{s}_{jh} = \tilde{s}(B_j, B_h)$, has been designed to find the smallest "gap" between hyperboxes and takes the following form:

$$\tilde{s}(B_j, B_h) = \min_{i=1..n} (\min([1 - f(v_{hi} - w_{ji}, \gamma_i)], [1 - f(v_{ji} - w_{hi}, \gamma_i)])) \tag{12}$$

   The characteristic features of this similarity measure are:

   (a) $\tilde{s}_{jj} = 1$
   (b) $0 \leq \tilde{s}_{jh} \leq 1 - \tilde{s}_{jh} = 1$ if there is any overlap between hyperboxes B_h and B_j
   (c) $\tilde{s}_{jh} = \tilde{s}_{hj}$—a degree of similarity of B_h to B_j is equal to a degree of similarity of B_j to B_h

   The properties (c) and (a) lead to a symmetrical similarity matrix with ones on its diagonal.

3. The third similarity measure between hyperboxes B_h and B_j, $\hat{s}_{jh} = \hat{s}(B_j, B_h)$, takes into account the maximum possible distance (on every dimension basis) between hyperboxes and takes the following form:

$$\hat{s}(B_j, B_h) = \min_{i=1..n} (\min([1 - f(w_{hi} - v_{ji}, \gamma_i)], [1 - f(w_{ji} - v_{hi}, \gamma_i)])) \tag{13}$$

   The characteristic features of this similarity measure are:

   (a) $0 \leq \hat{s}_{jj} \leq 1 - \hat{s}_{jj} = 1$ only if hyperbox B_j is a point and decreases with increasing size of B_j
   (b) $0 \leq \hat{s}_{jh} \leq \min(\hat{s}_{jj}, \hat{s}_{hh}) \leq 1$
   (c) $\hat{s}_{jh} = \hat{s}_{hj}$—a degree of similarity of B_h to B_j is equal to a degree of similarity of B_j to B_h

   The properties (c) and (a) lead to a symmetrical similarity matrix with values less or equal to one on its diagonal.

The illustration of respective similarity measures for a case of two hyperboxes in a two dimensional space is shown in Fig. 2.

Before progressing to the description of the agglomerative learning algorithms let us make few remarks concerning the above definitions. Strictly speaking (11) should not be referred to as a *similarity measure* since it

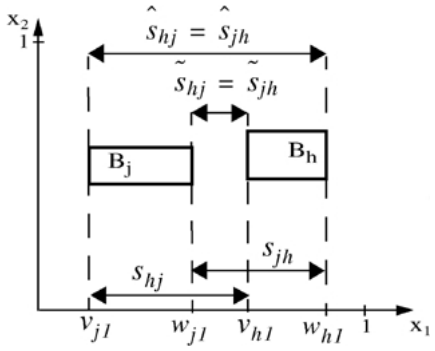

Figure 2. Graphical illustration of hyperbox similarity measures given by (11)–(13) which are proportional to the distances shown in a sense that the shorter the distance the higher the respective hyperbox similarity value.

does not satisfy the symmetry condition ($s_{jh} \neq s_{hj}$) as specified in the definition of similarity (dissimilarity) measures [14] applicable for vectors. It is also quite often the case that the term *similarity measure* is used when referring to measures between two vectors while the *proximity measure* term is used when referring to "proximity" (similarity) of clusters or sets of vectors. Although we are aware of these distinctions and the above definitions will be applied to hyperboxes representing both clusters and individual data patterns, in the remaining of the paper we will use the term similarity measures when referring to (11)–(13).

The similarity measures introduced above will now be used in the agglomerative process where in each step of the procedure two most similar hyperboxes (according to one of these measures) are aggregated to form a new hyperbox fuzzy set.

3.2. *Agglomerative Algorithm Based on Full Similarity Matrix (AGGLO-SM)*

The proposed training algorithm begins with initialization of the min points matrix $\boldsymbol{V}$ and the max points matrix $\boldsymbol{W}$ to the values of the training set patterns lower $\boldsymbol{A}^l$ and upper $\boldsymbol{A}^u$ limits respectively. Labels for a set of hyperboxes generated (initialised) in this way are assigned using the labels given in the training data set class$(B_k) = d_k$, for all $k = 1, \ldots, N$. If the GFMM neural network was to be used at this stage the resulting pattern recognition would be equivalent to the nearest neighbour method with (3) acting as a distance (similarity) measure for finding nearest neighbours.

In the next step a similarity matrix $\boldsymbol{S}$ is calculated using one of the similarity measures defined above. This similarity matrix is asymmetrical for similarity measure (11) and symmetrical for similarity measures (12) and (13). This fact has implications on how to decide whether a pair of hyperboxes are to be aggregated.

For the symmetrical similarity matrix among all possible pairs of hyperboxes (B_k, B_l) the pair (B_h, B_j) with maximum similarity value s_{jh} is sought. This can be expressed as:

$$\tilde{s}_{jh} = \max(\tilde{s}_{kl}) \quad \text{for all } k = 1 \ldots m-1, \; l = k+1 \ldots m \tag{14}$$

or

$$\hat{s}_{jh} = \max(\hat{s}_{kl}) \quad \text{for all } k = 1 \ldots m-1, \; l = k+1 \ldots m \tag{15}$$

for $\boldsymbol{S}$ derived from (12) and (13) respectively.

For the asymmetrical similarity matrix derived using (11) the selection of a pair of hyperboxes (B_h, B_j) to be aggregated is made by finding the maximum value from (a) the minimum similarity values $\min(s_{kl}, s_{lk})$; or (b) the maximum similarity values $\max(s_{kl}, s_{lk})$ among all possible pairs of hyperboxes (B_k, B_l). This can be expressed as:

$$s_{jh} = \max(\min(s_{kl}, s_{lk})) \quad \text{for all } k = 1 \ldots m-1, \; l = k+1 \ldots m \tag{16}$$

or

$$s_{jh} = \max(\max(s_{kl}, s_{lk})) \quad \text{for all } k = 1 \ldots m-1, \; l = k+1 \ldots m \tag{17}$$

Once the B_h and B_j have been selected for aggregation before the aggregate B_h and B_j operation is carried out, check if the following tests are passed:

(a) the overlap test (see [10] for details of hyperbox overlap test)
After temporarily aggregating hyperboxes B_h and B_j check if the newly formed hyperbox does not overlap with any of the hyperboxes representing different classes. If it does, take another pair of hyperboxes for potential aggregation.
What is interesting about the similarity measure (12) is the fact that it can be used for determining if two hyperboxes overlap. As shown in the previous

section $\tilde{s}_{jh} = 1$ if there is an overlap between hyperboxes B_h and B_j. Since we do not need to find out for which dimension the smallest overlap occurs, as it was required in the incremental version of the training algorithm, the similarity measure (12) can therefore be conveniently used for the overlap test purposes.

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

$$(\max(w_{ji}, w_{hi}) - \min(v_{ji}, v_{hi})) \leq \Theta \quad \text{for all } i = 1 \ldots n \tag{18}$$

(c) a test for the minimum similarity threshold ($0 \leq s_{\min} \leq 1$)

$$s_{jh} \geq s_{\min} \tag{19}$$

(d) a test for the class compatibility

$$\text{if class}(B_h) = 0 \quad \text{then aggregate } B_h \text{ and } B_j$$
$$\text{else}$$
$$\text{if class}(B_j) = \begin{cases} 0 \Rightarrow \text{aggregate } B_h \text{ and } B_j \\ \qquad \text{class}(B_j) = \text{class}(B_h) \\ \text{class}(B_h) \Rightarrow \text{aggregate } B_h \text{ and } B_j \\ \text{else} \Rightarrow \text{take another pair of} \\ \qquad \text{hyperboxes} \end{cases} \tag{20}$$

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

$$v_{ji}^{new} = \min\left(v_{ji}^{old}, v_{hi}^{old}\right) \quad \text{for each } i = 1, \ldots, n \tag{21}$$

$$w_{ji}^{new} = \max\left(w_{ji}^{old}, w_{hi}^{old}\right) \quad \text{for each } i = 1, \ldots, n \tag{22}$$

(b) remove B_h from a current set of hyperbox fuzzy sets (in terms of neural network shown in Fig. 1 it would mean a removal of the h-th second layer node).

(c) update the similarity matrix $\boldsymbol{S}$ by removing the h-th row and column and updating entries in the j-th row and column representing newly aggregated hyperboxes using v_{ji}^{new} and w_{ji}^{new}.

The above described process is repeated until there are no more hyperboxes that can be aggregated.

3.3. *The Second Agglomerative Algorithm (AGGLO-2)*

High computational complexity of the above described algorithm is mainly due to the need for calculating and sorting the similarity matrix $\boldsymbol{S}$ containing the similarity values for all possible pairs of hyperboxes. While for a given training set and the parameters Θ and $s_{\min}$ this ensures that the outcome of the training is always the same (does not depend on the order of input data presentation) it can be prohibitively slow for very large data sets.

The second agglomerative algorithm presented here attempts to reduce the computational complexity by not using the full similarity matrix during the process of selection and aggregation of pairs of hyperboxes.

Similarly to the first training algorithm, the min points matrix $\boldsymbol{V}$ and the max points matrix $\boldsymbol{W}$ are initialized to the values of the training set patterns lower $\boldsymbol{A}^l$ and upper $\boldsymbol{A}^u$ limits respectively. The hyperboxes are labelled using the training data set labels $\text{class}(B_k) = d_k$, for all $k = 1, \ldots, N$.

Rather than calculating the similarity values between all possible pairs of hyperboxes, the algorithm is based on cycling through the current set of hyperboxes selecting them in turn for possible aggregation with the remaining $m - 1$ hyperboxes.

In the first step the hyperbox fuzzy set B_j is selected as a first candidate for aggregation and similarity values between B_j and the remaining $m - 1$ hyperboxes are calculated.

The similarity values are sorted in the descending order and B_h with the highest similarity value s_{jh} is selected for potential aggregation which similarly to (14)–(17) can be expressed as:

$$\tilde{s}_{jh} = \max(\tilde{s}_{jl}) \quad \text{for all } l = 1 \ldots m, l \neq j \tag{23}$$

for the similarity measure (12);

$$\hat{s}_{jh} = \max(\hat{s}_{jl}) \quad \text{for all } l = 1 \ldots m, l \neq j \tag{24}$$

for the similarity measure (13);

$$s_{jh} = \max(\min(s_{jl}, s_{lj})) \quad \text{for all } l = 1 \ldots m, l \neq j \tag{25}$$

or

$$s_{jh} = \max(\max(s_{jl}, s_{lj})) \quad \text{for all } l = 1 \ldots m, l \neq j \tag{26}$$

for the similarity measure (11).

Once the B_h and B_j have been selected for aggregation the remaining steps (i.e. the four tests and aggregation itself) are the same as in the algorithm based on the full similarity matrix.

If B_h and B_j fail on any of the four tests the hyperbox with the second highest similarity value is used for potential aggregation with B_j and the process is repeated until there are no more hyperboxes which could be aggregated with B_j or if the aggregation takes place.

After the first aggregation has been performed there will be only $m - 2$ hyperboxes for further processing. Now the next hyperbox is selected for aggregation and the similarity values with the remaining $m - 2$ hyperboxes are calculated and the above described process repeated.

The training stops when after cycling through a whole set of hyperboxes there has not been a single aggregation performed.

The agglomeration of hyperboxes can be controlled by specifying different values for the parameters Θ and $s_{\min}$ 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 using AGGLO-2 which is explicitly created when using the agglomerative procedure based on the full similarity matrix. A similar effect can be obtained when using the parameter Θ by starting with relatively small values of Θ initially allowing to create small hyperbox fuzzy sets, and increasing the value of Θ in subsequent steps with inputs to the next level consisting of the hyperbox fuzzy sets from the previous level.

3.4. Agglomerative Learning in Clustering and Classification

As mentioned in the introduction and reflected in the above agglomerative learning algorithms, the GFMM neural network combines the supervised and unsupervised approaches within a single training algorithm. It can be used in a pure clustering (none of the training data are labelled—$d_h = 0$ for all training patterns), pure classification (all training data are labelled) or hybrid clustering-classification (the training data is a mixture of labelled and unlabelled patterns) problems.

3.4.1. Clustering. In case of pure clustering a hierarchy of nested clusterings can be obtained using the agglomerative procedures starting with N clusters representing training data patterns and potentially ending with a single cluster after N agglomerations have been performed. Similarly to other hierarchical clustering procedures a specific clustering can be obtained by setting the appropriate value for $s_{\min}$ or using one of the cluster validity criteria [14].

3.4.2. Classification. The GFMM neural network for classification belongs to a class of classification methods (i.e. nearest neighbour, unpruned decision trees, neural networks with a sufficient number of hidden nodes etc.) which have the capacity to learn the training data set perfectly. However, since the real world data to be classified are usually noisy or distorted in some way the classifier with zero resubstitution error rate would also model the noise and often produce class boundaries which are unnecessarily complex. It would also generally perform rather badly on unseen data.

The agglomerative learning procedures described in this paper, if applied to the full training data set, would result in creation of a neuro-fuzzy classifier which would have the zero resubstitution error rate. Therefore, in order to avoid overfitting and achieve good generalisation performance a suitable hyperbox fuzzy sets pruning procedure has to be used.

3.4.2.1. Hyperbox Fuzzy Sets Pruning Procedures. The hyperbox fuzzy sets pruning procedures which we have used with the agglomerative learning algorithms are based on assessing the contribution of individual hyperbox fuzzy sets to the performance of the GFMM classifier carried out during a 2-fold cross-validation or multiple 2-fold cross validation.

Two types of pruning approaches have been adopted which can be summarised as follows:

(a) after splitting the data set into training and validation sets and completing the training, perform a validation procedure by finding out how many of the validation set data patterns have been correctly classified and misclassified by each of the hyperbox fuzzy sets. Retain in the final classifier model only the hyperbox fuzzy sets which classify at

least the same number of patterns correctly as incorrectly.

(b) by performing a multiple cross-validation procedure estimate the minimum cardinality of a hyperbox fuzzy set for which the hyperbox fuzzy set should still be retained in the final GFMM classifier model. 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 rejected.

4. Simulation Results

The simulation experiments covering pure clustering, pure classification and hybrid clustering classification problems, have been carried out for a number of artificially created and real data sets taken from the repository of machine learning databases and some of them are reported in the sections below. The emphasis is put on illustration of strong and weak points of the agglomerative learning procedures and the flexibility of applying both on-line and agglomerative learning during different stages of designing a GFMM classifier for various data sets.

The following experiments have been divided into two groups: the first concerning a clustering problem and the second group considering various classification problems. In the following sections the training times for various data sets have been quoted but they should be used as an indication of the relative performance of the two agglomerative learning algorithms since no attempts for optimizing the programs have been made. All the simulations have been carried out in the MATLAB environment on a computer with an Intel Celeron 650 MHz processor and Microsoft Windows 2000 operating system.

4.1. Clustering

First a potential resistance of an agglomerative scheme to noise and outliers will be illustrated using a pure clustering problem. The two dimensional example used here consisted of two relatively dense clusters each consisting of 50 data patterns generated uniformly between [0.2 0.3] for the first cluster and [0.5 0.6] for the second cluster and additional 50 data patterns uniformly distributed in the input space representing noise. The hyperboxes have been initialised to the individual data points so the algorithm started with 150 hyperboxes. The data and clusters formed using (14)–(17) are shown in Fig. 3. The fact that the clusters are formed in the densest areas first with additional information about clusters cardinality has been used for filtering out outliers and the noisy data. In the example from Fig. 3 in order to remove noise, a very simple approach based on removing hyperboxes with a number of elements below a certain fixed level has been used. However, a number of different techniques falling into a cluster validation domain discussed in [14] could have been used instead but this is outside the scope of this paper.

It has to be said that the main idea behind hierarchical clustering algorithms is to produce a hierarchy of nested clusterings instead of a single clustering. This type of algorithm has been especially used in social sciences or biological taxonomy. One potential advantage of the hyperbox representation is an easy access to the limits (range of values) for each dimension for clusters at given levels.

Such a full hierarchy of clusterings is shown in Fig. 4 on the right in a form of dendrogram. This dendrogram has been obtained for the AGGLO-SM learning procedure using Eq. (15). The single clustering which was produced at the similarity level 0.9 represented by the dashed line in Fig. 4 is shown in Fig. 3(b). Figure 4 (left) also shows the aggregation levels at each of 150 aggregation steps using the AGGLO-SM algorithm and formulas (14) to (17). It illustrates an interesting phenomenon called *crossover* which refers to the fact that a new cluster can be formed at a higher similarity level than any of its components. This effect can be also observed on the dendrogram shown in Fig. 7. The opposite to the crossover is *monotonicity* which implies that each cluster is formed at a lower similarity level than any of its components. Only the agglomerative algorithm based on the similarity measure (13) satisfies the monotonicity criteria (Fig. 4—left, line a).

In order to determine the performance of the two agglomerative procedures the execution times for creating a full hierarchy of clusterings have been recorded which were: 27 seconds for the AGGLO-SM and 7 seconds for the AGGLO-2. The time for AGGLO-2 has been obtained for the procedure with 100 similarity levels (100 hierarchy levels) changing from 1 to 0 with a step 0.01. In this way a pseudo dendrogram could be created for AGGLO-2 representing a hierarchy of nested clusterings while significantly reducing the training time.

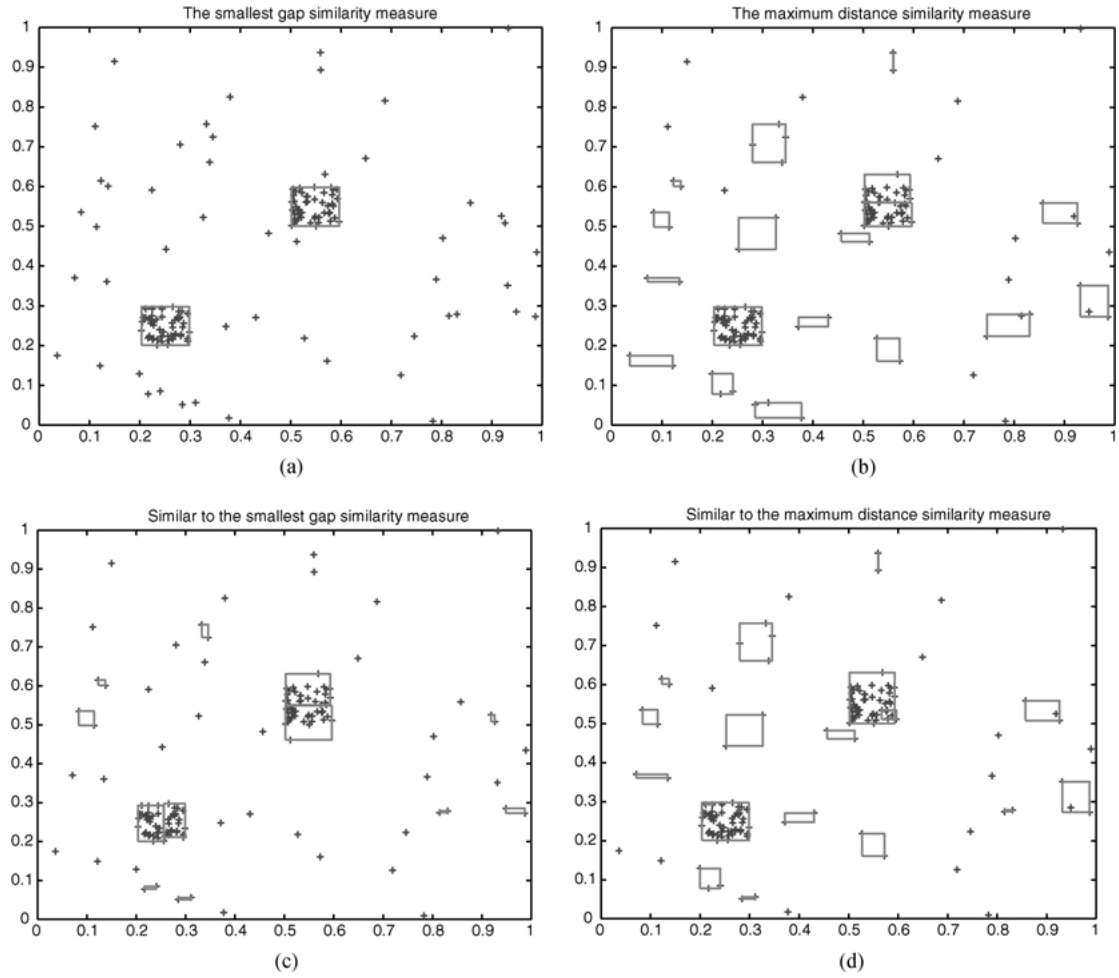

Figure 3. Illustration of a robust behaviour of an agglomerative algorithm for noisy data (a) using formula (14) and clustering obtained using formulas (b) (15) (c) (17) and (d) (16).

Carrying out various experiments, it has been found that agglomerative algorithms using formulas (14) and (17) are similar to the conventional single link algorithm and agglomerative algorithms using formulas (15) and (16) are similar to the conventional complete link algorithm.

4.2. Classification

Though agglomerative algorithms are usually applied for generating hierarchies of nested clusterings, as illustrated in the previous example, the following examples concern the use of the two proposed agglomerative learning procedures for five non-trivial data sets representing different classification problems.

The first two 2 dimensional, synthetic data sets represent cases of nonlinear classification problems with highly overlapping classes and a number of data points which can be classified as outliers or noisy samples. Using two dimensional problems also offer a chance of visually examining the effects of applying different similarity measures. 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, 17].

The other three data sets have been obtained from the repository of machine learning databases (http://www.ics.uci.edu/∼mlearn/MLRepository.html) and concern the problems of classifying iris plants (IRIS data set), three types of wine (Wine data set) and radar signals used to describe the state of ionosphere

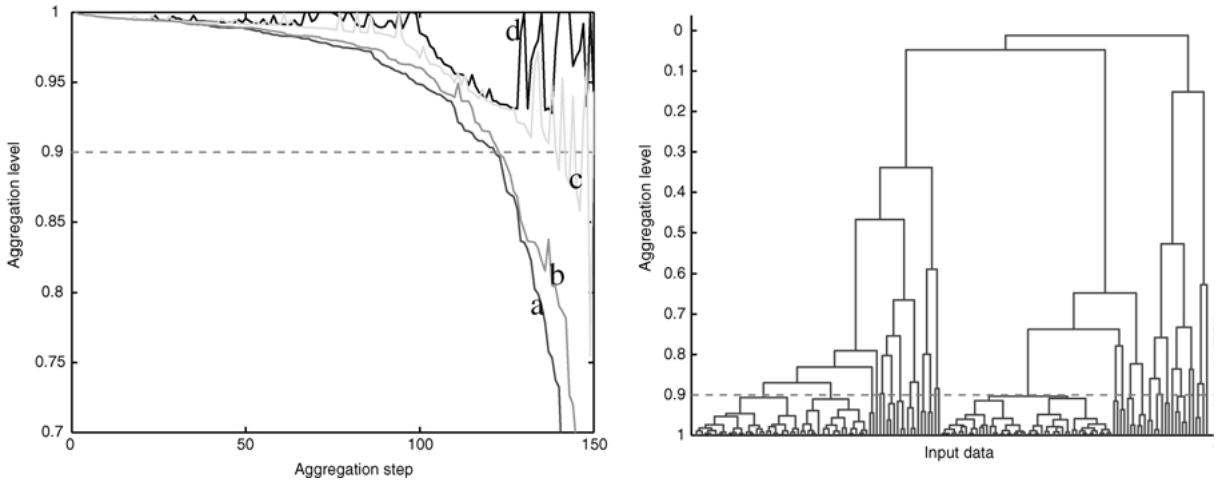

Figure 4. Left: Aggregation levels during the hyperbox aggregation process for clustering data using (a) (15), (b) (16), (c) (17) and (d) (14). Right: Dendrogram for the same clustering data obtained for (15). The clusters at the similarity level 0.9 from the dendrogram (dashed line) are shown graphically at Fig. 3(b). The similarity level of 0.9 is also highlighted by the dashed line in the left plot and applies to line a.

(Ionosphere data sets). The repository also contains the details of these data sets with some statistics and experimental results. 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.

Table 1. The sizes of data sets used in classification experiments.

			No. of data points		
Data set	No. of inputs	No. of classes	Total	Train	Test
Normal mixtures	2	2	1250	250	1000
Cone-torus	2	3	800	400	400
IRIS	4	3	150	75	75
Wine	13	3	178	90	88
Ionosphere	34	2	351	200	151

Table 2. Testing error rates (%) for four well known classifiers and five data sets.

	Classifier			
Data set	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
Ionosphere	1.99	3.97	7.28	2.65

4.2.1. Classification of the Normal Mixtures Data. The normal mixtures data has been introduced by Ripley [17]. The training data consists of 2 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 training set and an independent testing set of 1000 samples drawn from the same distribution are available at http://www.stats.ox.ac.uk/∼ripley/PRNN/.

The agglomerative learning procedures had been used during the 2-fold cross validation repeated 100 times and the cardinality of the hyperboxes to be retained in the final model was estimated. The cardinality for the normal mixtures data for all similarity measures was found to be 4. The agglomerative procedures were subsequently applied to the full training set and all the hyperboxes with 4 or less input data inside were removed. The resulting hyperboxes and misclassification rates for the testing set are shown in Fig. 5. The classification performance is surprisingly good with 3 out of 4 procedures resulting in error rates very close to the Bayes classifier optimal error rate of around 8%.

Another experiment carried out for this data set involved the investigation of the difference in the training times for the AGGLO-SM, the AGGLO-2 and a combination of the on-line learning algorithm and the AGGLO-SM. As it can be seen in Table 3 there is a large

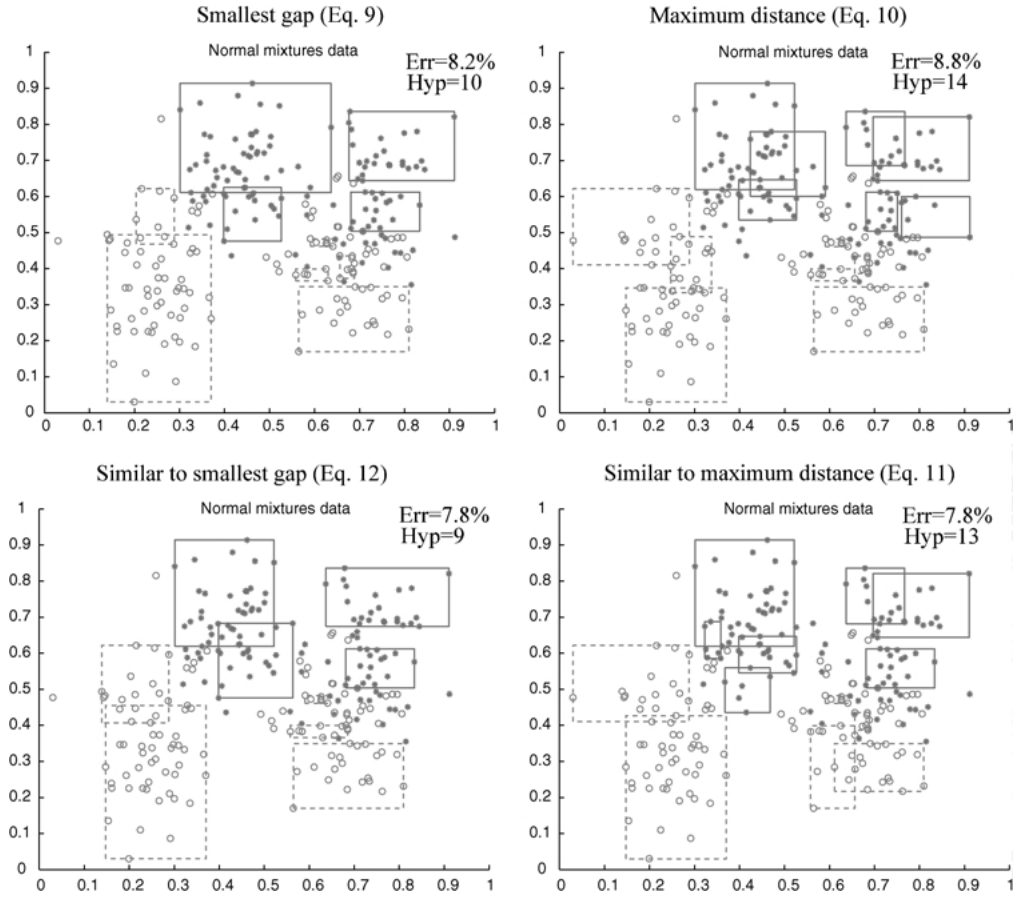

Figure 5. The training data and hyperboxes created for the normal mixture data set using the AGGLO-SM and all similarity measures. The performance for the testing set (Err) and the total number of hyperboxes retained after the pruning procedure (Hyp) are also shown.

difference in the training time between the AGGLO-2 and the AGGLO-SM, though both may seem very fast in comparison to the training times of some neural networks with backpropagation learning algorithm. In order to speed up the training times of the AGGLO-SM, the on-line learning can be used for an initial data clustering in order to reduce the number of inputs to be processed by the agglomerative procedure. By increasing the value of the parameter Θ used within on-line learning algorithm to control the maximum size of the

Table 3. Training times and classification performance of the AGGLO-SM (using (17)), the AGGLO-2 (using (26)) and combined on-line + AGGLO-SM learning algorithms for normal mixtures data.

			Online + AGGLO-SM				
Training algorithm	AGGLO-2	AGGLO-SM	$\Theta = 0.01$ (223)	$\Theta = 0.02$ (175)	$\Theta = 0.05$ (101)	$\Theta = 0.1$ (51)	$\Theta = 0.2$ (23)
Training time (s)	24 (s)	143 (s)	128 (s) (18 + 110)	77 (s) (14 + 63)	27 (s) (8 + 19)	7.6 (s) (4 + 3.6)	1.8 (s) (0.2 + 1.6)
Testing set error rate (%)	7.9	7.8	7.8	8.0	8.4	10	14.3

The numbers in brackets under the value of parameter Θ represent the number of hyperboxes returned from the on-line stage of learning and passed as inputs to the AGGLO-SM. The numbers in brackets in the training time row show the split training times for the on-line and the AGGLO-SM learning algorithms respectively.

hyperboxes which can be created, the training times can be significantly reduced. However, the performance of the GFMM classifier generated in this way suffered as also shown in Table 3.

4.2.2. Classification of the Cone-Torus Data. The cone-torus data set 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 testing set consisting of further 400 samples drawn from the same distribution are available at http://www.bangor.ac.uk/ ∼mas00a/.

As for the normal mixtures data set, the agglomerative learning procedures had been used during the 2-fold cross validation repeated 100 times and the cardinalities of the hyperboxes to be retained in the final model were estimated. The results of the testing for the AGGLO-2 learning algorithm are shown in Table 4. The upper part of Table 4 contains the results of the testing after the pruning procedure. As mentioned earlier in the paper the basic idea behind pruning is to remove hyperboxes representing noisy data and improve the generalisation performance. Once such "noisy" hyperboxes have been removed, it is possible that some of the remaining hyperboxes could be further aggregated. This is illustrated in the bottom part of Table 4 where the GFMM classifier has been retrained but only using the hyperboxes retained after initial pruning. The created hyperboxes during the pruning and retraining procedure are shown in Fig. 6.

As shown in Table 4, the retraining after pruning resulted in a significant reduction of the number of hyperboxes while not affecting significantly the classification performance. If anything a slight improvement of the performance in three cases out of four was observed.

We have yet again observed a much faster training for the AGGLO-2 algorithm which was just over 68 seconds for the smallest "gap" similarity measure in comparison to the training time for the algorithm based on full similarity matrix (AGGLO-SM) which for the same similarity measure was in excess of 455 seconds.

4.2.3. The Other Classification Data Sets. In comparison to results for IRIS, Wine and Ionosphere data sets obtained for incremental learning presented in [10], the agglomerative algorithms resulted in a very similar recognition performance (e.g. the recognition rates for the classifiers based on (14) and (23) are shown in Table 5) and generally fewer hyperboxes were required to encode the data. It is due to the earlier mentioned feature of creating clusters (hyperboxes) in the densest areas first and ability to discard clusters representing small number of input data. The results are also much more stable due to the removing of the dependency on the order of the training data presentation and need for resolving of undesired overlaps.

While for the two dimensional data sets it was very easy to visualise the created clusters by plotting the hyperboxes directly in the input pattern space, it is not

Table 4. Application of the AGGLO-2 learning algorithm to the cone-torus data set.

		Similarity values formulas			
		Eq. (23)	Eq. (26)	Eq. (25)	Eq. (24)
Pruning based on an estimation of the cardinality of the Hyperboxes to be retained in the final model	Training time (s)	68.43	54.25	71.25	112.72
	No of hyperboxes	30	33	30	31
	Cardinality of retained hyperboxes	>3	>3	>3	>1
	Training error (%)	11	11.25	8.5	9.5
	Testing error (%)	12.25	11	12	12.75
Pruning (as above) + retraining using only the hyperboxes retained after the pruning	No of hyperboxes	17	18	22	21
	Training error (%)	10.75	11	8.5	8.25
	Testing error (%)	10.5	11	11.5	13.25

Table 5. Summary of classification results for IRIS, Wine and Ionosphere data sets for both the AGGLO-SM (using (17)) and the AGGLO-2 (using (26)) learning algorithms.

Data set	Training algorithm	Training time (s)	No. of hyperboxes	Classification error for the training set (%)	Classification error for the testing set (%)
IRIS	AGGLO-2	1.29	4	2.67	4
	AGGLO-SM	6.02	3	2.67	4
Wine	AGGLO-2	5.21	6	1.14	1.14
	AGGLO-SM	27.8	4	1.14	2.67
Ionosphere	AGGLO-2	53.63	118	0	3.31
	AGGLO-SM	455.65	117	0	3.31

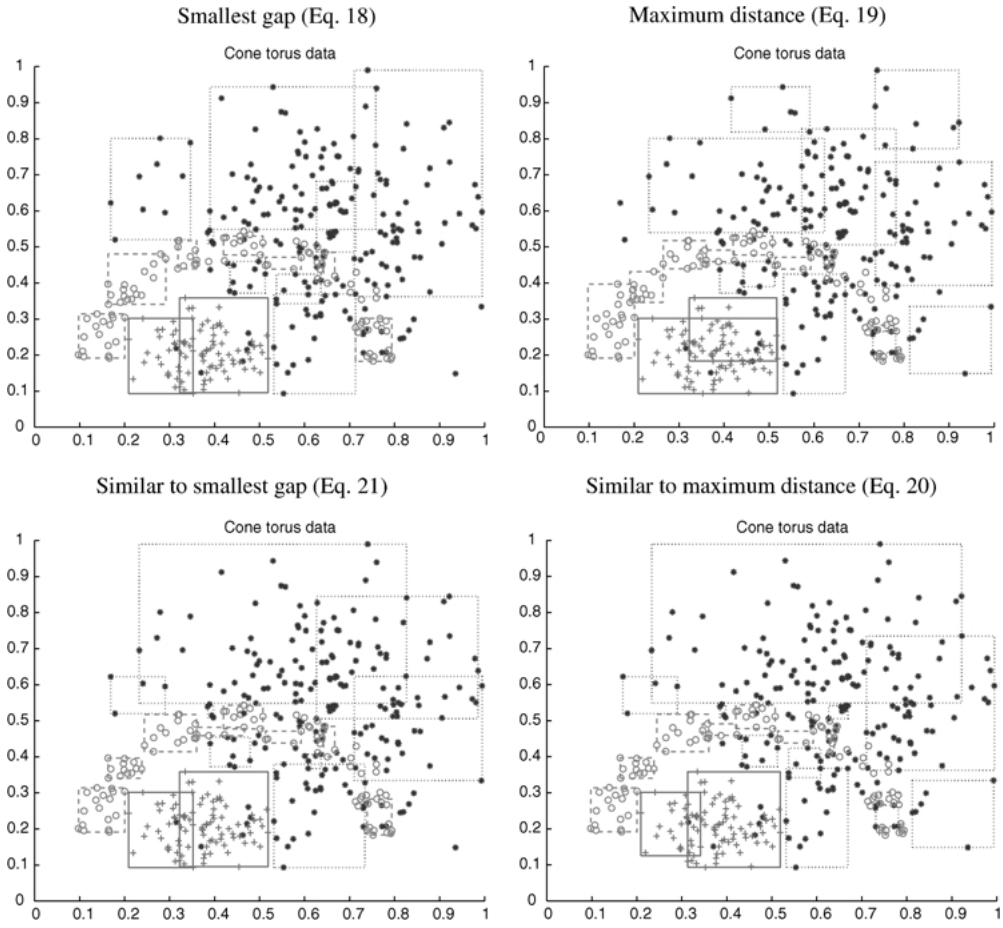

Figure 6. The result of the GFMM training using the AGGLO-2 and all similarity measures for the cone-torus data. The hyperboxes shown have been created by using the pruning procedure and retraining (see Table 4).

possible for higher dimensional problems. Although, hyperbox representations are much more transparent than some fully connected neural networks with nonlinear transfer functions since the hyperbox min and max points could be relatively easily interpreted by experts, the visualization of clusterings of high dimensional data in a form of dendrogram is a very attractive feature of the agglomerative algorithms. An

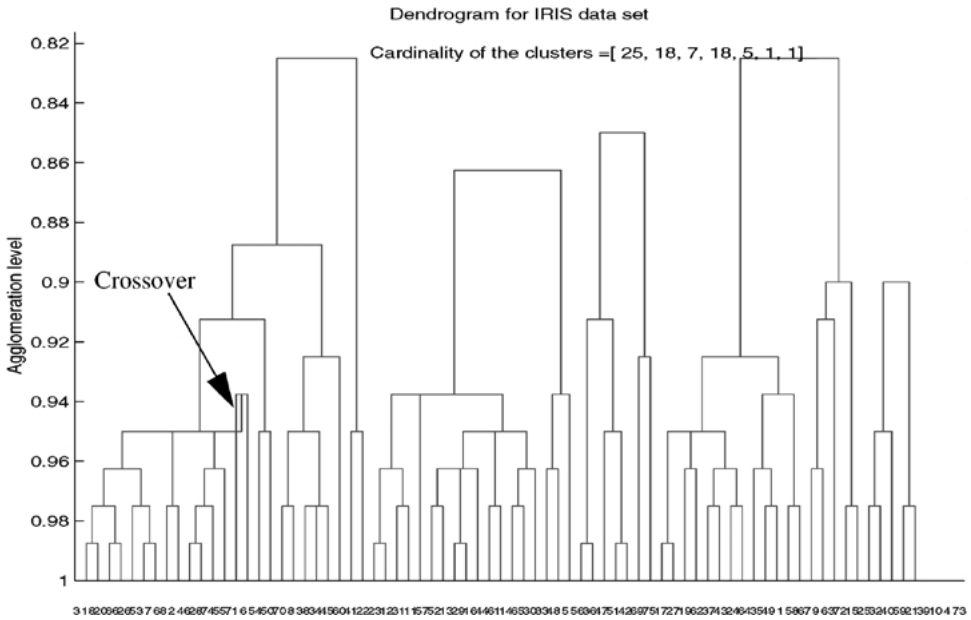

Figure 7. Dendrogram for IRIS data set obtained using the AGGLO-SM with formula (16).

example of a dendrogram for IRIS data set is shown in Fig. 7.

5. Summary and Conclusions

Two agglomerative learning algorithms utilising hyperbox fuzzy sets as cluster representatives have been presented in this paper. New similarity measures defined for hyperbox fuzzy sets have been introduced and used in the agglomerative schemes preserving the hybrid clustering/classification character of GFMM NN. A robust behaviour in presence of noise and outliers and insensitivity to training data presentation have been identified as main and the most valuable complimentary features to the previously proposed incremental learning.

A definite drawback of the agglomerative method based on the full similarity matrix is that for large data sets it can be very slow due to the size of the similarity matrix containing similarity values for all pairs of hyperboxes. This, however, can be overcome to a certain extent by using the incremental learning in the initial stages of GFMM training with a relatively small value of parameter Θ. The conducted experiments showed that a significant training time reduction could be obtained with increasing Θ. However, the vulnerability of including noise and outliers in the clusters formed at this stage was also increased.

In response to the potentially long training times for the agglomerative procedure based on the full similarity matrix, a second agglomerative learning algorithm has been developed which avoids calculating and sorting of the full similarity matrix. As a result, the computational complexity of the second algorithm has been substantially reduced leading to a significant reduction of training times. At the same time, the testing has shown no significant difference in the GFMM classification performance using any of the two agglomerative learning algorithms.

Acknowledgments

Research reported in this paper has been supported by the Nuffield Foundation grant (NAL/00259/G). The author would also like to acknowledge the helpful comments of the anonymous referees which contributed to the improvement of the final version of this paper.

References

1. M.H. Hassoun, *Fundamentals of Artificial Neural Networks*, Cambridge, MA: The MIT Press, 1995.
2. S. Mitra and K. Pal, "Self-Organizing Neural Network As a Fuzzy Classifier," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24, no. 3, 1994.

3. W. Pedrycz, "Fuzzy Neural Networks with Reference Neurons as Pattern Classifiers," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, 1992.
4. P.K. Simpson, "Fuzzy Min-Max Neural Networks—Part 1: Classification," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, 1992, pp. 776–786.
5. P.K. Simpson, "Fuzzy Min-Max Neural Networks—Part 2: Clustering," *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, 1993, pp. 32–45.
6. R.R. Yager and L.A. Zadeh (Eds.), *Fuzzy Sets, Neural Networks, and Soft Computing*, Van Nostrand Reinhold, 1994.
7. B. Gabrys, "Data Editing for Neuro Fuzzy Classifiers," in *Proceedings of the SOCO'2001 Conference*, Paisley, UK, 2001.
8. B. Gabrys, "Pattern Classification for Incomplete Data," in *Proceedings of 4th International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies KES'2000*, Brighton, vol. 1, 2000, pp. 454–457.
9. B. Gabrys and A. Bargiela, "Neural Networks Based Decision Support in Presence of Uncertainties," *J. of Water Resources Planning and Management*, vol. 125, no. 5, 1999, pp. 272–280.
10. B. Gabrys and A. Bargiela, "General Fuzzy Min-Max Neural Network for Clustering and Classification," *IEEE Trans. on Neural Networks*, vol. 11, no. 3, 2000, pp. 769–783.
11. J. Boberg and T. Salakoski, "General Formulation and Evaluation of Agglomerative Clustering Methods with Metric and Non-metric Distances," *Pattern Recognition*, vol. 26, no. 9, 1993, pp. 1395–1406.
12. H. Frigui and R. Krishnapuram, "Clustering by Competitive Agglomeration," *Pattern Recognition*, vol. 30, no. 7, 1997, pp. 1109–1119.
13. H. Frigui and R. Krishnapuram, "A Robust Competitive Clustering Algorithm with Applications in Computer Vision," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, 1999, pp. 450–465.
14. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, San Diego, CA: Academic Press, 1999.
15. K. Ozawa, "Classic: A Hierarchical Clustering Algorithm Based on Asymetric Similarities," *Pattern Recognition*, vol. 16, no. 2, 1983, pp. 201–211.
16. L.I. Kuncheva, *Fuzzy Classifier Design*, Heidelberg: Physica-Verlag, 2000.
17. B.D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.

Bogdan Gabrys received an MSc degree in Electronics and Telecommunication (Specialization: Computer Control Systems) from the Silesian Technical University, Poland in 1994 and a PhD in Computer Science from the Nottingham Trent University, UK in 1998.

Dr Gabrys now works as a Lecturer at the University of Paisley, Division of Computing and Information Systems. His current research interests include a wide range of machine learning and hybrid intelligent techniques encompassing data and information fusion, multiple classifier systems, processing and modelling of uncertainty in pattern recognition, diagnostic analysis and decision support systems. He published around 20 research papers in the areas of mathematical modelling, simulation, artificial neural networks, computational intelligence, soft computing, pattern recognition, decision support, and optimisation. Dr Gabrys has also reviewed for various journals, co-edited special issues of journals, chaired sessions and been on programme committees of a number of international conferences with the Computational Intelligence and Soft Computing theme.

Dr Gabrys is a corresponding person for a Key Node in the European Network on Intelligent Technologies for Smart Adaptive Systems (EUNITE) and committee member in the Research Theory & Development Group on Integration of Methods. He is a member of the Institute of Electrical and Electronics Engineers (IEEE) and the Institute for Learning and Teaching (ILT) in Higher Education.
gabr-ci0@paisley.ac.uk