DATA EDITING FOR NEURO FUZZY CLASSIFIERS

Bogdan Gabrys

Applied Computational Intelligence Research Unit
Division of Computing and Information Systems
University of Paisley, High Street, Paisley PA1 2BE
Scotland, United Kingdom
Tel: +44 (0) 141 848 3752, Fax: +44 (0) 141 848 3542
E-mail: gabr-ci0@paisley.ac.uk

1. ABSTRACT

In this paper we investigate the potential benefits and limitations of various data editing procedures when constructing neuro-fuzzy classifiers based on hyperbox fuzzy sets. There are two major aspects of data editing which we are attempting to exploit: a) removal of outliers and noisy data; and b) reduction of training data size. We show that successful training data editing can result in constructing simpler classifiers (i.e. a classifier with a smaller number and larger hyperboxes) with better generalisation performance. However we also indicate the potential dangers of overediting which can lead to dropping the whole regions of a class and constructing too simple classifiers not able to capture the class boundaries with high enough accuracy. A more flexible approach than the existing data editing techniques based on estimating probabilities used to decide whether a point should be removed from the training set has been proposed. An analysis and graphical interpretations are given for the synthetic, non-trivial, 2-dimensional classification problems.

2. INTRODUCTION

The main objective of a classifier design process is constructing as simple a classifier as possible with as good classification performance as possible.

The classifier design generally involves a learning procedure which based on a finite number of training data samples attempts to update the parameters of the classifier in such a way as to reduce the misclassification rate. Depending on the complexity of the classifier the error rate for the training data set (also known as the resubstitution error rate) can be reduced to zero. Examples of such classifiers include the nearest neighbour, unpruned decision trees, neural networks with a sufficient number of hidden nodes or unpruned neuro fuzzy classifiers to be discussed in this paper. However, since the real world data to be classified are usually noisy or distorted in some way due to errors in measurements, recording problems etc. the classifier attempting to reduce the resubstitution error rate to zero would also model the noise and often produce the class boundaries which are unnecessarily complex. It would also generally perform rather badly on unseen data. The above problems are known as the training data overfitting and the classifier generalisation ability.

Since the ultimate goal is low generalisation error one requires some procedures that would ensure/increase a chance of the good generalisation performance. One of the most common approaches to avoid overfitting is based on cross-validation where a training set is split into a number of disjoint sets which are then used separately for the training and model validation\generalisation error estimation. This process is then repeated many times in order to obtain reliable statistics.

While for many problems the generalisation error can be estimated through multiple cross-validation it does not solve one basic but very important problem: Which of the classifier models generated during the repeated training should be chosen as a final classifier? There is also another dilemma: How can one use as many of the training data samples and still produce a classifier with good generalisation performance?

Since the noisy training samples are the potential reason for poor generalisation performance, another approach could be to identify such data and remove them from the training set. Such approach is investigated in this paper through analysis of the usefulness of data editing techniques when designing a neuro-fuzzy classifier.

Data editing techniques have been a subject of several studies [3,11] associated with classifiers using the k-nearest neighbour rule. One of the commonly acknowledged disadvantages of k-nn classifiers is that they require the storage of a large number of samples and that finding k-nearest neighbours for such large training data sets can take too long to compute. However, it has also been observed that a subset of the training set is usually sufficient to approximate very well the decision boundaries. What is more, in cases when training data set contains outliers and noisy data, the use of all training samples in the k-nn classifier design usually lead to worse classification performance (due to overfitting) than when only a suitable subset of the training data is used. The data editing procedures have therefore been applied with the aims of increasing the computational efficiency, through reduction of the number of class prototypes, and improving the generalisation performance through

filtering out the outliers and noisy samples.

The general fuzzy min-max (GFMM) neural network for classification [6-8] which will be used in this paper belongs to a class of classification methods (similarly to the 1-nn method) which have the capacity to learn the training data set perfectly. In the type of the boundaries it creates and representation of the classification regions it is similar to decision trees [4], fuzzy rule based classifiers where the rules are learned directly from data [1,2,10], and other classification methods which use hyperboxes (or hyperrectangles) as the data cluster prototypes [12,14]. In all these methods a suitable pruning procedure has to be used in order to avoid overfitting and achieve good generalisation performance.

The data editing in connection with GFMM will be used as a method of preserving as many "good" training data samples as possible while attempting to remove samples which would cause the overfitting. Three different data editing approaches will be examined. The first two will be the adaptations of the rejection of the misclassified samples using the leave-one-out and n-fold cross-validation error estimation approaches [3,5,11]. The third one will be based on the identification and estimation of the importance of different data points from the training set which are consistently used in the generation of the classifiers during multiple n-fold cross-validation procedure.

The remaining of this paper is organized as follows. Section 3 will give an overview of the GFMM NN and the agglomerative learning algorithm. In Section 4 the description of the data editing approaches will be given. This will be followed by experimental results. And finally the conclusions will be presented in the last section.

3. AN OVERVIEW OF 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-point and its max-point. 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 pattern space. For more details concerning the on-line training algorithm please refer to [8] while the summary of the agglomerative learning procedure will follow later in this section. 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:

$$\boldsymbol{A}_h = [\boldsymbol{A}_h^l \;\; \boldsymbol{A}_h^u]$$

where $\boldsymbol{A}_h^l$ and $\boldsymbol{A}_h^u$ 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 $\boldsymbol{A}_h^l = \boldsymbol{A}_h^u$ the input represents a point in the pattern space.

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

$$B_j = \{\boldsymbol{V}_j, \boldsymbol{W}_j, b_j(\boldsymbol{A}_h, \boldsymbol{V}_j, \boldsymbol{W}_j)\} \tag{1}$$

for all j=1,2,...,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:

$$b_j(\boldsymbol{A}_h) = \min_{i=1..n}(\min([1 - f(a_{hi}^u - w_{ji}, \gamma_i)], [1 - f(v_{ji} - a_{hi}^l, \gamma_i)])) \tag{2}$$

where:

$$f(x, \gamma) = \begin{cases} 1 & if \quad x\gamma > 1 \\ x\gamma & if \quad 0 \le x\gamma \le 1 \\ 0 & if \quad x\gamma < 0 \end{cases}$$ - two parameter ramp

threshold function; $\gamma = [\gamma_1, \gamma_2, \ldots, \gamma_n]$ - sensitivity parameters governing how fast the membership values decrease; and $0 \le b_j(\boldsymbol{A}_h, \boldsymbol{V}_j, \boldsymbol{W}_j) \le 1$.

The membership values are used to decide whether the presented input pattern belongs to the class associated with the *j*-th hyperbox during the neural network operation stage.

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

$$c_k = \max_{j=1}^{m} b_j u_{jk} \tag{3}$$

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..p*, represent the degrees of membership of the input pattern in the *k*-th class.

Agglomerative learning

The agglomerative learning for GFMM [6] initializes the min **V** and max **W** matrices to the values of the training set patterns lower $\mathbf{A}^l$ and upper $\mathbf{A}^u$ limits respectively. The hyperboxes are then aggregated

sequentialy (one pair at a time) on the basis of the maximum similarity value calculated using the following similarity measure $\tilde{s}_{jh} = \tilde{s}(\boldsymbol{B}_j, \boldsymbol{B}_h)$ between B_h and B_j:

$$\tilde{s}_j(\boldsymbol{B}_h) = \min_{i=1..n}(min([1 - f(v_{hi} - w_{ji}, \gamma_i)], [1 - f(v_{ji} - w_{hi}, \gamma_i)]))$$

This similarity measure finds the smallest "gap" between B_h and B_j and resulting clustering algorithm is similar to the conventional single link algorithm [13]. Other similarity measures defined for hyperbox fuzzy sets are discussed in [6].

The hyperboxes with highest similarity value are only aggregated if:

a) the aggregation does not result in an overlap with any of the hyperboxes representing other classes;

b) newly formed hyperbox does not exceed the maximum allowable hyperbox size; and

c) the hyperboxes B_h and B_j form a part of the same class.

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

After the training of the GFMM using the agglomerative learning procedure the training set is learned perfectly and in order to avoid overfitting a hyperbox pruning procedure has to be applied. An alternative to the standard cross-validation procedure in form of the training data editing will now be described.

4. DATA EDITING PROCEDURES

The basic data editing procedures reported in the literature process the training data set with the aim of removing the data samples which contribute to the misclassification rate.

Following [15] let us first give the general description of the data editing procedure which can be applied with any classifier with different error estimate procedures.

1) Make a random partition of the data set, *R*, into *N* groups $R_1,...,R_N$.

2) Classify the samples in the set R_i, using a classifier designed using a union of the remaining *M* sets $(1 \le M \le N-1)$ as the training set and tested on R_i. Repeat *M* times taking different R_i each time. Let *S* be the set of misclassified samples found during all *M* runs.

3) Remove all misclassified samples from the data set to form a new data set, *R=R-S*.

4) If the last *I* iterations have resulted in no samples being removed from the training data set then terminate the algorithm, otherwise go to step 1.

Depending on the values of M and when the classifier in point 2 of the above procedure is the k-nn rule we can get the multiedit algorithm of Devijver and Kittler (*M=1)* [3], the Wilson's method of editing using leave-one-out approach (*M=N-1* where *N* in this case would be the number of samples) [16] etc.

Procedure 1: The first of the data editing procedures examined in this paper follows the above general description. It is based on an application of the 1-nn, 3-nn and 5-nn classification rules within a leave-one-out scheme. To make it suitable for the GFMM algorithm the membership function (2) acts as the similarity measure utilised for finding k-nearest neighbours. Please notice that the GFMM NN can be easily adapted for each of these schemes. In this case the hyperboxes represent individual training data points and the outputs of the second layer nodes can be treated as the classification values. The k maximum values from the second layer outputs are used together with the above classification rules to decide whether the input is classified correctly or not.

The following two data editing schemes are based on identifying either the misclassified points or the data points which have been used for generating the GFMM classifiers at any stage during a multiple cross-validation process. The cross-validation used here is a two-fold cross validation (M=1) where the training set is split into two equally sized (as much as possible) partitions R_1 and R_2 and the design of GFMM classifier is first carried out on R_1 and validated on R_2 and then the process is repeated for R_2 used as a design set and R_1 as a validation set. The two-fold cross-validation has been chosen due to our observation that the GFMM model generated strongly depends on having large sets both for the design and validation stages while the two sets should be as independent (non overlapping) as possible.

Procedure 2: The second data editing procedure is a slightly modified version of the general algorithm given at the beginning of this section. The main difference is that after points 1 and 2, the misclassified data are not removed at point 3 but only marked and the random splitting into two separate sets continues for the original training data set. The process can be repeated a fixed number of times (i.e. 100 times) or stopped if the last *I* iterations have not resulted in new samples being marked as misclassified. Only after the multiple cross-validation process is completed all the marked samples are removed and the GFMM trained for the edited training set. As it will be illustrated in the next section, this approach, which is similar to the multiedit algorithm, will result in forming homogenous clusters in the input space. However, as it has also been indicated in the literature removing all the misclassified samples using the above approach can sometimes result in dropping whole regions of a class and essentially overediting the training set. In terms of the GFMM classifier generated for such edited data set, due to overediting, the classification models have turned out to be too simple to accurately capture the decision boundaries.

Procedure 3: The third data editing procedure has been designed using an observation that some input data samples are more consistently used for generation of classifiers during the cross-validation process than others. It also applies to the misclassified samples whereas some of the samples will be misclassified consistently while others only occasionally. The approach based on the rejection of all the samples which have been misclassified at least ones during the multiple cross-validation does not attempt to take this fact into account in any way. In order to rectify this problem we propose an approach which will estimate the probability of every single point in the original training data set to be used in the generation of the hyperboxes during the multiple cross-validation. This probability is simply calculated as the ratio of the number of times ($\mathbf{N}_h$) an input $\mathbf{A}_h$ has been used in generation of a hyperbox which is retained in the classifier model after the validation to the total number of repetitions (**N**) of the two-fold cross validation. This can be expressed as:

$$P(\boldsymbol{A}_h) = \frac{N_h}{N} \tag{4}$$

Examples of the sorted probability distribution for the training sets of the two classification problems are shown in Fig. 3d and Fig. 4d. The small values of the probabilities indicate which of the input data points should be removed first when editing the training data set. The points with P=0 should always be removed while the points with P=1 should always be retained in the training set. In this paper a number of different levels in between have been tested and results reported but in the future a suitable procedure which would optimally select the probability value on the basis of the probability distribution will be investigated.

5. SIMULATION RESULTS

The results of applying the above editing procedures are illustrated on two 2-dimensional, synthetic classification problems. Both data sets have been selected because of the fact that they 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 data editing. 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 [9,11].

The first data set represents a normal mixtures data which have been introduced by Ripley [11] and is shown at Fig. 1. 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/.

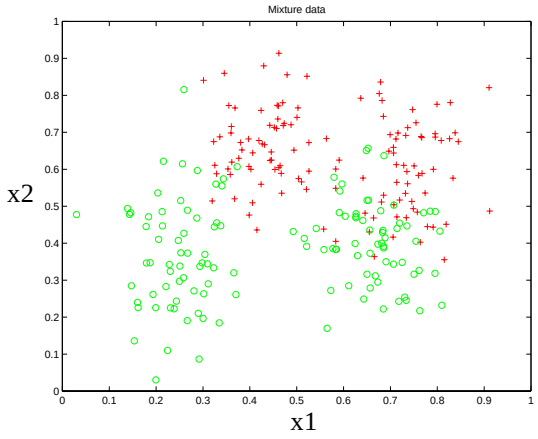

Figure 1: Normal mixture data set. Samples from the two different classes represented by '+' and 'o'.

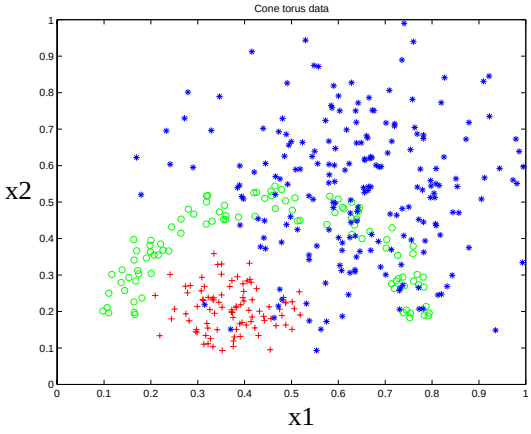

Figure 2: Cone-torus data set. Samples from the three different classes represented by '+', '', and 'o'.*

The second data set illustrated at Fig. 2 has been introduced by Kuncheva [9]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/.

The results of testing the data editing procedures introduced in the previous section for the above two data sets are shown in Table 2 and Table 3. Graphical illustrations for each of the tested editing procedures together with the hyperboxes created on the basis of the edited training sets are shown in Fig. 3 and Fig. 4. For

comparison purposes the classification performance of GFMM, when no editing has been used but only 2 fold cross-validation procedure employed, is illustrated in Table 1. For further comparisons of some standard editing techniques for the normal-mixtures and cone-torus data sets please refer to [9 section 3.3].

Data set	Misclassification rate for the training set [%]			Misclassification rate for the testing set [%]		
	Mean error	Standard deviation	Range of errors [min max]	Mean error	Standard deviation	Range of errors [min max]
Normal mixtures	13.48	1.54	[10 18.4]	9.64	1.03	[7.7 12.7]
Cone-torus	12.54	1.69	[8 17.5]	14.78	1.57	[10.5 19.5]

Table 1: GFMM classification results based on 2 fold cross-validation repeated 100 times (no editing).

As can be seen in the Table 2 and Table 3, Procedure 1 based on k-nn rule and leave-one-out method performed well in connection with GFMM classifier resulting in a reduction of the model complexity (i.e. generation of smaller number of hyperboxes) while improving generalisation performance. Examples of edited sets for this editing procedures are shown at Fig. 3a and Fig. 4a. Using larger k results in retaining input data forming more homogenous clusters and favouring classes with larger number of samples. Procedure 2 turned out to be too strong for our purposes and in both cases resulted in removing too many points near the boundaries of the classes in order for GFMM to capture those boundaries well. The examples of the edited sets and hyperboxes created when using Procedure 2 are shown at Fig. 3b and Fig. 4b.

The most promising results have been obtained for Procedure 3 which is based on finding probabilities of individual data points (please see Fig. 3d and Fig. 4d) describing how likely a point from the original training data set is to be used in formation of the hyperboxes defining the final GFMM classifier

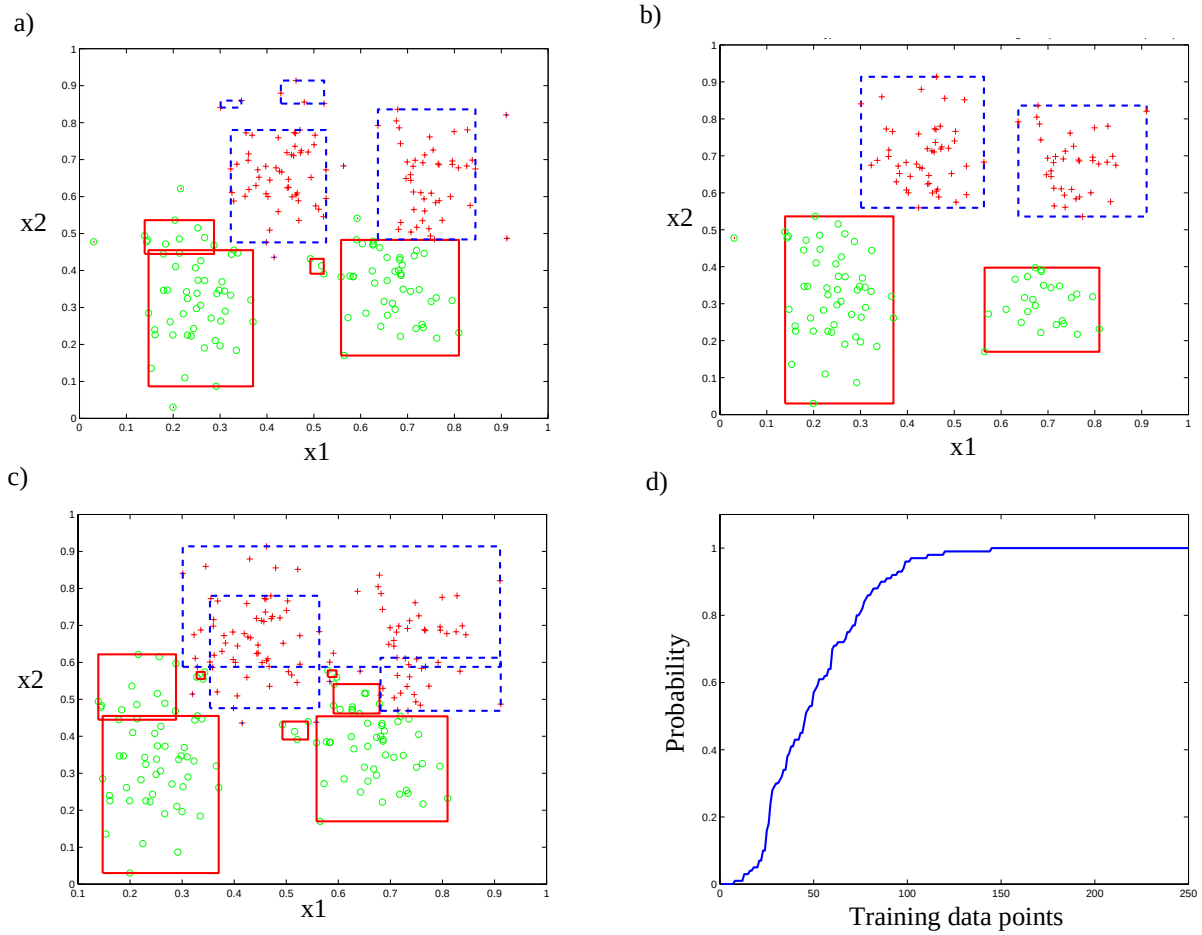

Figure 4: The edited training data sets and created hyperboxes for the normal mixtures data set. a) Procedure 1 for 3-nn classifier; b) Procedure 2 for 2-fold cross validation repeated 100 times and all misclassified samples rejected; c) Procedure 3 for 2-fold cross validation repeated 100 times and samples with P<0.1 rejected; d) sorted probability distribution for all the training data samples which have been used with Procedure 3

The results presented here are for 4 different values of P including the two extreme cases where only the points with P=0 are removed from the training set and only the points with P=1 are retained in the edited training set. The first case (points with P=0 were removed) showed signs of overfitting while the second extreme case (points with P<1 were removed) resulted in generating too simple classifiers. When we chose the case in between (points with P<0.5 were removed) the resulting classifiers exhibited a very good generalisation performance (better than the average case reported in Table 1) with substantially reduced classifier complexity in comparison to non-edited training set. For the normal-mixtures data set when the points with P<0.1 have been removed we have observed the classification performance very close to the absolute optimum of 8%. The optimal selection of P_{rem} such that all points with $P<P_{rem}$ are removed from the original training set and the resulting GFMM classifier exhibits the best generalisation performance will be the subject of further investigations which could involve the analysis of the reduction of complexity together with change of the training data error when gradually higher P_{rem} is selected.

6. CONCLUSIONS

An analysis of the usefulness and potential benefits and limitations of various data editing procedures when designing a GFMM classifier has been the main subject of this study. The data editing techniques have been used to remove some input data samples which otherwise would result in constructing too complex classifiers with poor generalisation ability. In this sense they can be viewed as an alternative to the commonly used cross-validation procedures. We have shown that some of the well known editing techniques which remove all the misclassified data can be too restrictive and result in overediting the training set. We have proposed an alternative which takes into account the fact that some input data points are more likely to be misclassified than

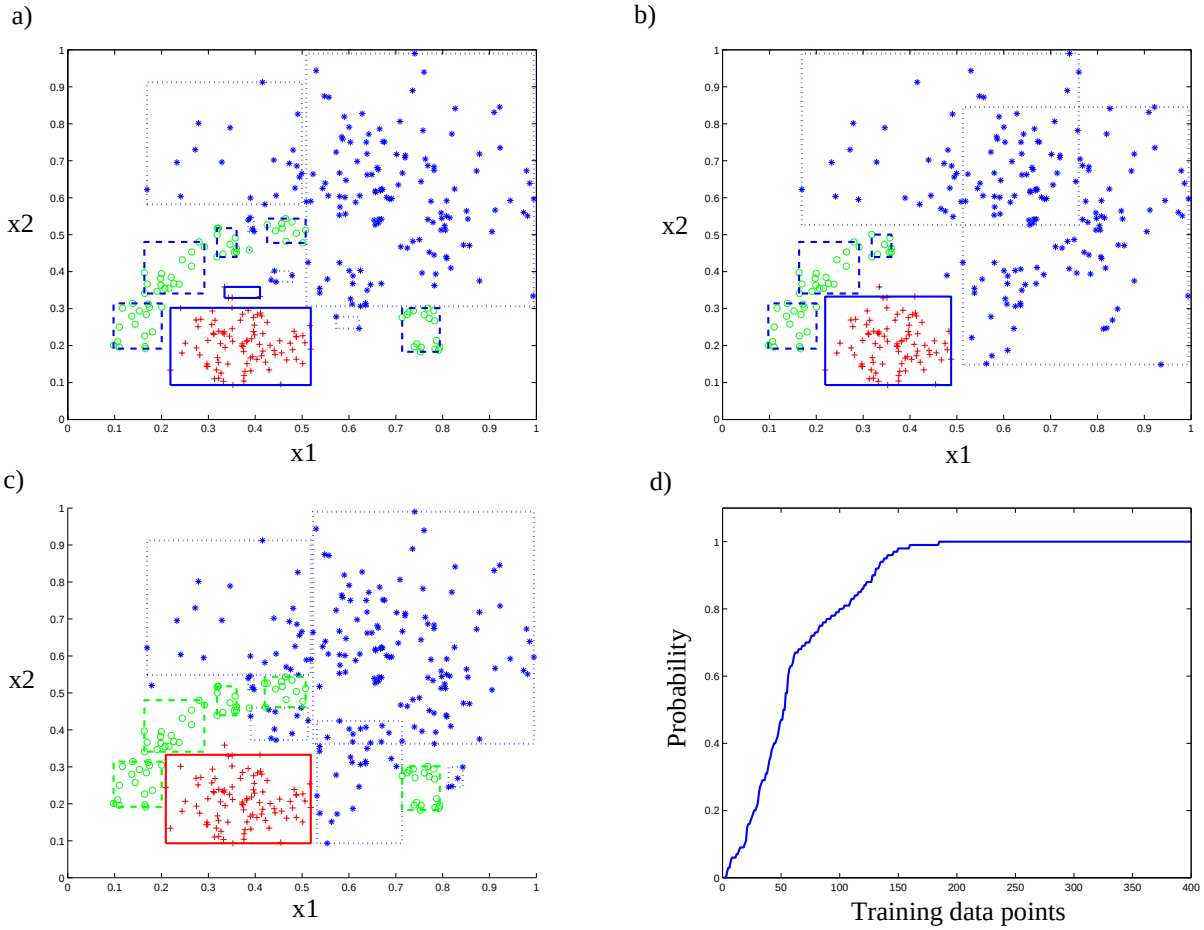

Figure 3: The edited training data sets and created hyperboxes for the cone-torus data set. a) Procedure 1 for 5-nn classifier; b) Procedure 2 for 2-fold cross validation repeated 100 times and all misclassified samples rejected; c) Procedure 3 for 2-fold cross validation repeated 100 times and samples with P<0.5 rejected; d) sorted probability distribution for all the training data samples which have been used with Procedure 3

Description of the data editing procedure		No of data points left (rejected)	No of hyperboxes created	Misclassification rate for the training set [%]	Misclassification rate for the testing set [%]
Not edited training set		250 (0)	37	0	12.1
Procedure 1: k-nn with leave-one-out	1-nn	213 (37)	18	8.0	10.5
	3-nn	207 (43)	16	12.0	8.9
	5-nn	209 (41)	11	12.4	9.0
Procedure 2: Rejection of all misclassified input samples		162 (88)	5	16.4	10.0
Procedure 3: Probability guided rejection procedure	P=0	243 (7)	33	2.0	10.7
	P<0.1	228 (22)	15	8.0	8.3
	P<0.5	205 (45)	8	11.6	8.4
	P<1	106 (144)	4	16.4	9.7

Table 2: The results of testing different data editing procedures for the normal mixtures data set.

Description of the data editing procedure		No of data points left (rejected)	No of hyperboxes created	Misclassification rate for the training set [%]	Misclassification rate for the testing set [%]
Not edited training set		400 (0)	55	0	15.0
Procedure 1: k-nn with leave-one-out	1-nn	327 (73)	20	8.75	13.5
	3-nn	330 (70)	18	10.0	13.75
	5-nn	318 (82)	13	12.75	13.5
Procedure 2: Rejection of all misclassified input samples		289 (111)	8	17.25	17.5
Procedure 3: Probability guided rejection procedure	P=0	397 (3)	50	0.75	14.5
	P<0.1	382 (18)	36	4.25	13.75
	P<0.5	349 (51)	15	9.25	13.0
	P<1	216 (184)	6	18.0	16.5

Table 3: The results of testing different data editing procedures for the cone-torus data set.

others. By empirically estimating probabilities describing which of the training data points are most likely to be removed from the training set we showed that removal of the points with the smallest probabilities can result in constructing simpler and better performing classifiers.

Acknowledgments

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

REFRENCES

[1] Abe, S. and M.Lang, "A Method for Fuzzy Rules Extraction Directly from Numerical Data and Its Application to Pattern Classification", *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 1, pp.18-28, 1995

[2] Berthold, M., "Fuzzy Models and Potential Outliers", in Proceedings of NAFIPS-99, pp. 532-535, IEEE Press, 1999

[3] Devijver, P.A. and J.Kittler, *Pattern Recognition: A Statistical Approach*,Prentice-Hall Int., 1982

[4] Duda, R.O., P.E.Hart, D.G.Stork, *Pattern Classification*, John Wiley & Sons, 2000

[5] Ferri, F.J., J.V.Albert, and E.Vidal, "Considerations About Sample-Size Sensitivity of a Family of Edited Nearest-Neighbour Rules", *IEEE Trans. on Syst., Man, and Cyber. - PART B: Cybernetics*, vol. 29, no. 4, pp. 667-672, 1999

[6] Gabrys, B., "Agglomerative Learning for General Fuzzy Min-Max Neural Network", to appear in *the Proceedings of IEEE NNSP'2000 workshop*, Sydney, December 2000

[7] Gabrys, B., A.Bargiela, "Neural Networks Based Decision Support in Presence of Uncertainties", *ASCE J. of Water Resources Planning and Management*, Vol. 125, No. 5, pp. 272-280, 1999

[8] Gabrys, B., A.Bargiela, "General Fuzzy Min-Max Neural Network for Clustering and Classification", accepted for publication in *IEEE Transactions on Neural Networks*, 2000

[9] Kuncheva, L.I., *Fuzzy Classifier Design*, Physica-Verlag Heidelberg, 2000

[10] Nauck, D. and R.Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data", *Fuzzy Sets and Systems*, vol. 89, no. 3, pp. 277-288, 1997

[11] Ripley, B.D. *Pattern Recognition and Neural Networks*, mbridge University Press, 1996

[12] Simpson, P.K., "Fuzzy Min-Max Neural Networks - Part 1: Classification", *IEEE Trans on Neural Networks*, vol. 3, no. 5, pp.776-86, September 1992

[13] Theodoridis, S., K.Koutroumbas, *Pattern Recognition*, Academic Press, 1999

[14] Tschichold-Gurman, N., "The neural network model RuleNet and its application to mobile robot navigation", *Fuzzy Sets and Systems*, vol. 85, pp.287-303, 1997

[15] Webb, A., *Statistical Pattern Recognition*, Arnold (Hodder Headline Group), 1999

[16] Wilson, D., "Asymptotic Properties of NN Rules Using Edited Data", *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 2, no. 3, pp.408-421, 1972