ELSEVIER

European Journal of Operational Research 123 (2000) 16–28

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/orms

Theory and Methodology

Fuzzy rule based classification with *FeatureSelector* and modified threshold accepting

V. Ravi [*,1], H.-J. Zimmermann

Lehrstuhl für Unternehmensforschung, RWTH, Templergraben 64, D-52056, Aachen, Germany

Received 8 September 1998; accepted 22 December 1998

Abstract

This paper highlights the need to reduce the dimension of the feature space in classification problems of high dimensions without sacrificing the classification power considerably. We propose a methodology for classification tasks which comprises three phases: (i) feature selection, (ii) automatic generation of fuzzy if–then rules and (iii) reduction of the rule base while retaining its high classification power. The first phase is executed by using *FeatureSelector*, a software developed solely for feature extraction in pattern recognition and classification problems. This is for the first time that the *FeatureSelector* is used as a preprocessor for rule based classification systems. In the second phase, a standard fuzzy rule based classification system is modified and invoked with the most important features extracted by the *FeatureSelector* as the new set of features. In the third phase, a modified threshold accepting algorithm (MTA), proposed elsewhere by the authors (Ravi et al., Fuzzy Sets and Systems, forthcoming) is used for minimizing the number of rules in the classification system while guaranteeing high classification power. The number of rules used and the classification power are taken as the objectives for this multi objective combinatorial global optimization problem. The methodology proposed here has been successfully demonstrated for two well-known problems (i) the wine classification problem, which includes 13 feature variables in its original form and (ii) the Wisconsin breast cancer determination problem, which has 9 feature variables. In conclusion, the results are encouraging as there is no remarkable reduction in the classification power in both the problems, despite the fact that some features have been deleted from the study by resorting to feature selection. Also, the MTA outperformed the original threshold accepting algorithm for the test problems considered here. The authors suggest that classification problems having higher feature dimensions can be solved successfully within the framework of the methodology presented here. The high classification powers obtained for both the problems when working with less feature variables than the original number is the significant achievement of this study.

Keywords: Fuzzy sets; Feature selection; Fuzzy if–then rule bases; Modified threshold accepting; Meta-heuristics; Multi objective combinatorial optimization problem

* Corresponding author. Tel.: +49-241-804-628; fax: +49-241-8888-168.

E-mail address: vravi@or.rwth-aachen.de (V. Ravi).

[1] On deputation from Computer Center, Indian Institute of Chemical Technology, Uppal Road, Hyderabad - 500 007, Andhra Pradesh, India.

0377-2217/00/$ - see front matter
PII: S0377-2217(99)00090-9

1. Introduction

Among the earliest applications of fuzzy set theory are knowledge based systems. The core of such systems are fuzzy 'if–then' rules. Most of these systems derive these fuzzy 'if–then' rules from human experts [10]. Several methods have been proposed to generate these fuzzy 'if–then' rules directly from numerical data. Kosko [15] presented methods based on neural networks to achieve this goal. Later, Ishibuchi et al. [10] came out with a general purpose, easy-to-understand methodology to generate such rules from the numerical data and then they applied a genetic algorithm to determine a compact rule set with a high classification power [11]. Some of the machine learning based works which also generate the fuzzy if–then rules are Yuan and Shaw [25], Ichihashi et al. [13] and Yuan and Zhuang [26]. A version of fuzzy-ID3 algorithm which induces fuzzy decision trees is proposed in [25]. In [13], a neuro-fuzzy ID3 algorithm for inducing decision trees and incremental learning is presented and Yuan and Zhuang [26] propose a fuzzy genetic algorithm for generating fuzzy classification rules. Then MIT GmbH, Aachen, Germany, started marketing a software *WINROSA* [24] which uses statistical methods to automatically generate fuzzy 'if–then' rules from numerical data. However, all the above mentioned works differ from that of [10,11] in many ways.

In what follows, the partition of a pattern space means its granularity. The generation of fuzzy if–then rules from numerical data involves (i) the fuzzy partition of a pattern space into fuzzy subspaces and (ii) the determination of fuzzy if–then rules for each fuzzy partition [10,11]. Next follows the classification phase, where either the training data or the test data are classified using the fuzzy if–then rules generated. The performance of such a classification system depends on the choice of a fuzzy partition. If a fuzzy partition is too coarse, the performance may be low, because many patterns may be misclassified. On the other hand, if a fuzzy partition is too fine, many fuzzy if–then rules can't be generated due to the lack of training patterns in the corresponding fuzzy subspaces. Therefore, the choice of a fuzzy partition is very important. In their earlier paper, Ishibuchi et al. [10] have proposed distributed fuzzy rules, where fuzzy rules corresponding to both coarse partitions and fine partitions of a fuzzy subspace are taken into account simultaneously. For example, a two-dimensional pattern space gives rise to 90 $(=2^2 + 3^2 + 4^2 + 5^2 + 6^2)$ fuzzy if–then rules, assuming that each pattern space is divided into 6 partitions at the most. Thus, they have considered all 5 rule tables corresponding to all the partitions simultaneously. Also by considering all the fuzzy partitions simultaneously, the above mentioned difficulty in choosing an appropriate partition is circumvented.

The main drawback of this approach, however, is that the number of fuzzy if–then rules becomes enormous for classification problems with high-dimensional pattern spaces [12] such as the wine classification problem [9] which has 13 feature variables. In such cases, the method proposed in Refs. [10,11] cannot be applied straightaway, as it would increase exponentially the total number of rules from which an efficient rule set is to be derived. For example, if 5 partitions are used for each of the 13 feature variables, the total number of possible rules would become $2^{13} + 3^{13} + 4^{13} + 5^{13} \cong 1.29 \times 10^9$. This is the case for the distributed representation of fuzzy if–then rules. Even if single rule tables were considered, for 5 partitions, the total number of possible rules would become $5^{13} \cong 1.22 \times 10^9$. Thus the search space becomes unmanageably large and finding a compact rule set with high classification power from such a search space would consume enormous amount of computational time and memory as well. Thus it is sensible and logical to resort to a mechanism by which the number of candidate rules become less even for a problem whose feature space dimension exceeds, say 5, with a reasonable number of partitions. This drawback was first observed by Ishibuchi et al. [12] where they observed that a problem with even 4 features and 6 partitions can exert considerable pressure on CPU time and memory. To circumvent this problem, Ishibuchi et al. [12] have proposed an improvement to their original model and introduced a method where the fuzzy if–then rules with a small number of antecedent conditions are generated as

candidate rules. They observed that their proposed way of reducing the number of candidate rules works well.

However, the authors are of the opinion that it is still not a complete remedy because it is not difficult to find problems where rules with small number of antecedent conditions are intractable. The authors also feel that there is a paramount aspect viz., feature selection inherent in this kind of problems which needs to be addressed appropriately. This calls for the development of other alternative methods.

Thus it is meaningful to look for any unimportant features and remove them from the classification process much before the fuzzy if–then rules generation phase starts. This results in reduced computational time and memory requirement and a viable classification system with manageable number of features. Thus feature selection is an important component of classification algorithms specially in dealing with problems having a large number of features. To achieve this objective, in this paper, *FeatureSelector*, a software developed by Strackeljan et al. [22,23], was used as a preprocessor. It is for the first time that the *FeatureSelector* is used in fuzzy rule based classification systems. It filters the unimportant features and gives the best combinations of features in a decreasing order of classification power. Once feature selection phase is over, the fuzzy if–then rule generation phase takes over, where the model of Ishibuchi et al. [11] is modified slightly and used. The final phase is the formulation and solution of a multi-objective combinatorial global optimization problem with the classification power and the number of rules as the objectives. A MTA proposed elsewhere by the authors [21] is used to solve this problem.

The structure of the paper is as follows. Section 2 contains an overview of the *FeatureSelector* software. Section 3 presents briefly the fuzzy classification system and Section 4 formulates the multi-objective combinatorial global optimization problem and then describes, in detail, the MTA giving the flowchart. Section 5 presents the details of the numerical examples solved and then discusses the results obtained and Section 6 concludes the paper.

2. FeatureSelector – An overview

That feature selection is a quintessential component of classification tasks has been recognised as early as 1973, when Kittler and Young [14] surveyed the then existing feature selection procedures and proposed a new feature selection algorithm based on Karhunen–Loeve expansion. Then Chang [6] applied dynamic programming to this problem. The most recent work is that of Bradley et al. [4] who proposed a linear programming formulation of the feature selection problem. These are only a few of the existing feature selection algorithms. The *FeatureSelector*, which is a plug-in for the commercial software on soft computing and intelligent data analysis, *Data-Engine* [7], has its roots in fuzzy logic. As the name suggests, it selects the significant features and acts as a pre-processor for the classification problems with very high feature dimensions. For instance, the acoustic analysis of vibration signals in the time and frequency domain generates a large number of features and makes the reduction of the dimensionality an absolute necessity [22,23]. Until now the testing and diagnostic tasks have not been satisfactorily automated because after the completion of the learning phase defining a set of features is necessary which describes the pattern as unambiguously as possible. In such cases, an expert establishes the features by filtering the data. Automation of feature selection requires measures of quality, based on which the respective combination of features can be examined for suitability. *FeatureSelector* is a contribution to integrate the feature extraction process and a classification algorithm. The aim is to limit the need for a human expert only to the supervised learning phase. Because *DataEngine* is a powerful software tool which not only contains different methods for soft classification and clustering but also has the interesting feature to integrate additional functionalities by way of Plug-Ins, it was used as the platform for the development of the *FeatureSelector* [22,23].

The problem of feature selection lies in selecting the best subset Y of n features i.e. $Y = \{y_i | i = 1, 2, \ldots, n\}$ from the total set $X = \{x_i | i = 1, 2, \ldots, n'\}$, where $n < n'$. The development of a

classification system is frequently accomplished by separately appraising the feature selection step and the inherent classification step. However, the *FeatureSelector* combines the two steps because it has the advantage that the selected features are well suited for the given classifier. Since the applied optimizing criteria are then directly related to the selected classification algorithm, the selected features satisfy the mathematical conditions for this algorithm in each case.

The following measures are used by the *FeatureSelector*

1. Reclassification error: On the basis of a classified learning sample for which unambiguous class assignment has been performed for each random sample during the learning phase, a measure of appraisal has been defined by reclassifying the learning sample with the respective classification algorithm. Thus, at each step the program calculates how many objects from the training set can be classified correctly when the considered feature combination is used. For this, the class membership of each object is required. Reclassification rate is defined as the ratio of the number of correctly classified samples to the total number of samples. Reclassification error is defined as (1.0-Reclassification rate). Thus, a reclassification rate of 1.0 means that all samples were assessed correctly, which implies that the reclassification error is zero. Hence, such a feature combination is suitable for the classification task.
2. In addition to the reclassification error, the quality criterion 'Distance' assesses the certainty with which an object is assigned to a class. For this, the difference between memberships of that object to a class with highest membership value and the class with the next highest membership value is calculated. This criterion is based on the concept that a feature combination is especially suitable if the arrangement in the feature space allows as great a distance as possible to arise between the classes. It is introduced to create one more criterion for ranking features with the same reclassification rate.

The algorithm for the assessment of the reclassification rate is used first. If two different combinations give rise to identical errors, the combination for which the quality criterion 'distance' is greater is given a better overall assessment. One important feature of this software is the flexibility it provides to the user regarding the choice of the number of significant features that are to be selected. Once a number has been chosen by the user, the *FeatureSelector* provides various combinations of features in the increasing order of the reclassification error. More details of this software can be found in Strackeljan et al. [22,23].

3. A fuzzy classification method with fuzzy if–then rules

Following the notation of [10,11], let the pattern space be two-dimensional (i.e. there are two features in the feature space) and given by the unit square $[0,1] \times [0,1]$ for simplicity of notation. For more details of the extension to the case of higher dimensions, the reader is referred to [11]. Suppose that m patterns $X_p = (x_{p1}, x_{p2})$, $p = 1, 2, \ldots, m$, are given as training patterns from M classes (where $M \ll m$): Class 1 (C1), Class 2 (C2), …, Class M (CM). The problem is to generate fuzzy if–then rules that divide the pattern space into M disjoint classes.

As in [10,11], let us suppose that each axis of the pattern space is partitioned into K fuzzy subsets $\{A_1^K, A_2^K, \ldots, A_K^K\}$, where A_i^K is the ith fuzzy subset and the superscript K is attached to indicate the number of fuzzy subsets on each axis (see Fig. 1). Thus, K denotes the grid size of a fuzzy partition. Since all the fuzzy rule tables are considered simultaneously in the distributed representation of fuzzy rules [10], it is essential to keep the subscript K to denote the fuzzy rules corresponding to a fuzzy rule table with K partitions, $K = 2, \ldots, L$. A symmetric triangular membership function is used for A_i^K, $i = 1, 2, \ldots, K$.

$$\mu_i^K(x) = \max\{1 - |x - a_i^K| / b^K, 0\}, \quad i = 1, 2, \ldots, K,$$

where

$$a_i^K = (i-1)/(K-1), \quad i = 1, 2, \ldots, K, \qquad b^K = 1/(K-1). \tag{1}$$

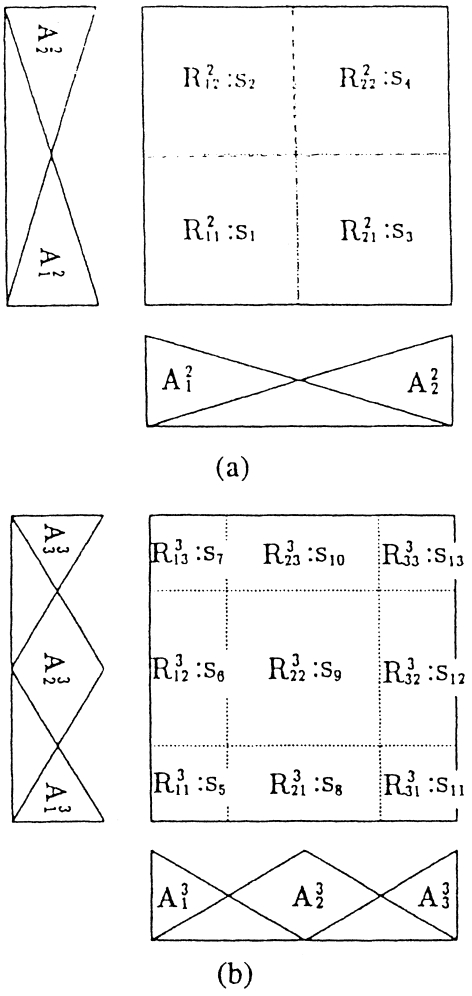

Fig. 1. Labels and indexes of fuzzy if–then rules (a) $k=2$, (b) $k=3$ [11].

For classification problems involving M classes and 2 features, a fuzzy if–then rule corresponding to K^2 fuzzy subspaces has the following structure.

Rule R_{ij}^K: If x_{p1} is A_i^K and x_{p2} is A_j^K then X_p belongs to Class C_{ij}^K,

$$\text{with } CF = CF_{ij}^K, \quad i = 1,2,\ldots,K \text{ and } j = 1,2,\ldots,K, \tag{2}$$

where $R_{ij}^K$is the label of the fuzzy if–then rule, $A_i^K$and A_j^K are the triangular fuzzy subsets on the unit interval [0,1], C_{ij}^K is the consequent (i.e. one of the M classes) and $CF_{ij}^K$is the grade of uncertainty of the fuzzy if–then rule. $C_{ij}^K$and $CF_{ij}^K$of the fuzzy if–then rule in Eq. (2) are determined by the following procedures [10,11].

Procedure 1. Generation of Fuzzy if–then rules:

1. Calculate β_{CT} for each class $T = 1,2,\ldots,M$ as $\beta_{CT} = \sum_{X_p \in CT} \mu_i^K(x_{p1}) \bullet \mu_j^K(x_{p2})$, where β_{CT} is the sum of compatibility of X_p's in class T to the fuzzy if–then rule R_{ij}^K in (2) and the symbol '$\bullet$' represents the *product* operator in the original works of Ishibuchi et al. [10,11]. However, in this paper, it stands for one of the following t-norms/operators viz. (1) product operator, (2) *min* operator, (3) γ-operator (compensatory *and*), (4) fuzzy *and* and (5) a convex combination of *min* and *max* operators. The same is valid for procedure 2 also.
2. Find class X (CX) such that

$$\beta_{CX} = \max\{\beta_{C1}, \beta_{C2}, \ldots, \beta_{CM}\}. \tag{3}$$

   If β_{CX} is not unique in Eq. (3) or all the β_{CT}'s are zero, the consequent C_{ij}^K of the fuzzy if–then rule R_{ij}^K can not be determined uniquely. In this case let C_{ij}^K be φ. On the other hand, if β_{CX} is unique in Eq. (2), then C_{ij}^K is determined as CX in Eq. (3).
3. If β_{CX} is unique, CF_{ij}^K is determined as follows:

$$CF_{ij}^K = (\beta_{CX} - \beta) \Big/ \sum_{T=1}^{M} \beta_{CT}, \quad \text{where } \beta = \sum_{T=1, T \neq X}^{M} \beta_{CT}/(M-1). \tag{4}$$

In other words, the consequent C_{ij}^K is determined as class X (CX) which has the largest sum of $\mu_i^K(x_{p1}) \bullet \mu_j^K(x_{p2})$ over the M classes in Eq. (2). Fuzzy if–then rules with φ in the consequent part are dummy rules that have no effect on fuzzy inference for classifying new patterns. If there is no pattern in the fuzzy subspace $A_i^K \times A_j^K$, a dummy rule is generated for this fuzzy subspace because all the β_{CT}'s become zero in this case. The certainty factor CF_{ij}^K is specified by Eq. (4). □

Let S^K be the set of generated K^2 fuzzy if–then rules given by $S^K = \{R_{ij}^K | i = 1,2,\ldots,K;\ j = 1,2,\ldots,K\}$. That is, S^K is the rule set corresponding to

the $K \times K$ fuzzy rule table. In the approach of distributed fuzzy if–then rules [10,11], multiple fuzzy if–then rules are used simultaneously. Let the set of all fuzzy if–then rules corresponding to $K = 2, 3, \ldots, L$ partitions be S_{ALL} which is

$$\begin{aligned} S_{\mathrm{ALL}} &= S^2 \cup S^3 \cup \cdots \cup S^L \\ &= \{R_{ij}^K |\ i = 1, 2, \ldots, K;\ j = 1, 2, \ldots, K \text{ and} \\ &\qquad K = 2, 3, \ldots, L\}, \end{aligned} \tag{5}$$

where L is an integer to be specified depending on the classification problem. By taking $K = 2, 3, \ldots, L$ in Procedure 1 above, $2^2 + 3^2 + \cdots + L^2$ fuzzy if–then rules are generated. Let S be a subset of S_{ALL}. The main objective is to find a compact rule set S with very high classification power by taking recourse to a combinatorial global optimization formulation with multiple objective functions. This is described in detail later.

The procedure for the classification of a new pattern into any of the M classes is presented as follows in Procedure 2 [10,11].

Procedure 2. Classification of new patterns $X_p = (x_{p1}, x_{p2})$

1. Find α_{CT} for each class T $(T = 1, 2, \ldots, M)$ as

$$\begin{aligned} \alpha_{CT} = \max\{&\mu_i^K(x_{p1}) \bullet \mu_j^K(x_{p2}) \times CF_{ij}^K | CF_{ij}^K \\ &= CT \text{ and } R_{ij}^K \in S\}. \end{aligned} \tag{6}$$

2. Find the $X(CX)$ such that $\alpha_{CX} = \max\{\alpha_{C1}, \alpha_{C2}, \ldots, \alpha_{CM}\}$. If α_{CX} is not unique or all the α_{CT}'s are zero, then the classification of X_p is rejected (i.e. X_p is left as unclassifiable pattern), else X_p is classified as $X(CX)$.

4. The modified threshold accepting for solving the multi-objective combinatorial optimization problem

The model of Ishibuchi et al. [11] has been modified as follows: Firstly, the decision making part of the rule generation and classification phases, where the *product* and the *min* operators were used, is modified slightly, in that several other aggregators have been tried instead. They are: (i) compensatory *and* (γ-operator), (ii) fuzzy *and* and (iii) a convex combination of *min* and *max* operators. Secondly, instead of using a genetic algorithm to solve the multi objective combinatorial global optimization problem, in this paper, another meta-heuristic viz. a MTA is devised and adapted.

With procedure 1, the fuzzy if–then rules corresponding to $K = 2, 3, \ldots, L$ are generated from the training patterns X_p, $p = 1, 2, \ldots, m$. Thus the rule set S_{ALL} is obtained. Now the problem is to construct a compact rule subset from S_{ALL}, which has a high classification power. Thus, there are two objectives in the problem: (i) maximize the number of correctly classified patterns and (ii) minimize the number of fuzzy if–then rules. These objectives can be used to formulate a multi objective combinatorial global optimization problem as follows [11]:

$$\begin{aligned} &\text{Maximize} \quad \mathrm{NCP}(S) \text{ and Minimize } |S| \\ &\text{subject to} \quad S \subseteq S_{\mathrm{ALL}}, \end{aligned}$$

where NCP(S) is the number of correctly classified patterns by S and $|S|$ is the cardinality of S (i.e. the number of fuzzy if–then rules in S). Assuming that the unknown preference (utility) function of the decision maker is modelled by combining his/her two objectives additively by the weights W_{NCP} and W_S, the compromise solution is determined by using the following objective function:

$$\begin{aligned} &\text{Maximize } f(S) = W_{\mathrm{NCP}} \cdot \mathrm{NCP}(S) - W_S \cdot |S| \\ &\text{subject to } S \subseteq S_{\mathrm{ALL}}. \end{aligned} \tag{7}$$

It should be kept in mind, however, that the algorithm used is a heuristic and that, depending on the initial feasible solution the best solutions reported here may not be efficient.

In [10], it was observed correctly that, in general, the classificatory power of the system is more important than its compactness. Accordingly, the weights above are specified as $0 < W_S \ll W_{\mathrm{NCP}}$. In this paper, the values of W_{NCP} and W_S are taken as 10 and 1, respectively, following [10]. For the sake of computations, the coding of the rules has been done as follows. The rule set is represented as an array, where each component corresponds to one rule in the $N(= 2^2 + 3^2 + \cdots + L^2)$ possible rules. Each rule has two main states: (i) rule is a dummy

rule and (iii) rule is a non-dummy rule. The non-dummy rules, which ultimately classify the patterns, are again having two states in a given candidate solution vector. These are: (i) rule belongs to S, (ii) rule does not belong to S. These two states are represented by 1 and -1 respectively. Since dummy rules have no influence on the fuzzy inference in classification phase (i.e. Procedure 2), they should be excluded from S. Hence, they are represented by zero [11].

In the recent literature, global optimization techniques such as, Simulated Annealing, Tabu Search, Genetic Algorithms, Threshold Accepting are all grouped in one category and called meta-heuristics [19]. Threshold Accepting, proposed by Dueck and Scheuer [8], is a variant of the original simulated annealing algorithm in that the acceptance of a new move or solution is determined by a deterministic criterion rather than a probabilistic criterion. Dueck and Scheuer [8] showed through numerical experimentation that threshold accepting is superior to simulated annealing for solving combinatorial global optimization problems. In this paper, the threshold accepting algorithm has been modified and adapted to the problem already described in the previous section.

4.1. Modified threshold accepting algorithm

The algorithm presented in the following is written for a minimization problem. Hence, to maximize the objective function in Eq. (7) we need to multiply it with -1. The flowchart of the algorithm is presented in Fig. 2. Let $a[i]$, $i = 1, 2, \ldots, N$, be the candidate solution vector, where N is the total number of possible rules in the system. Thus N becomes 90 $(= 2^2 + 3^2 + \cdots + 6^2)$ in the case of two-dimensional feature space with each feature partitioned into 6 fuzzy subsets. Let m be the number of classes. Let each component of $a[i]$ represent a fuzzy rule.

Step (i): Generate an initial feasible solution randomly using a uniform random number generator. Here we adopted a biased probability method, according to which, presence of a rule is assigned a chance of 0.05 and absence of a rule is assigned a chance of 0.95. That is,

$$a[i] = \begin{cases} 1, & \text{if } U \geqslant 0.95, \\ -1, & \text{if } U < 0.95. \end{cases} \tag{8}$$

The generation of a solution in a random way is limited to the initial solution only. Also, 0.95 is not fixed for both the problems and for all the cases. Thus, for some cases, it is taken as a value in the range [0.9, 0.995]. The best solution obtained after performing many simulations with different initial solutions, is reported for each case in Tables 1–4.

Step (ii): Calculate the value of the objective function in Eq. (7) for this initial feasible solution and store it in f_i. Let $itr = 0$, $thresh = 0.035$, $eps = 0.35$, $thrtol = 10^{-8}$, $acc = 10^{-6}$, $old = 9999$ and $itrmax = 200$.

Step (iii): Start the global iteration: $itr = itr + 1$.

Step (iv): Start the inner iteration which is essentially a neighbourhood search. To accomplish this a deterministic procedure is devised and employed as follows. Starting from rule 1 (i.e. initially $i = 1$) and going up to rule N, check whether each rule is a dummy rule or not. If a rule is not dummy, then its status is changed, i.e. if a non-dummy rule is present in the candidate rule set, then it is forced to become absent from the rule set and vice versa. This results in a neighbouring solution of the original one. Thus, if $a[i] \neq 0$, then $a[i] = -a[i]$ Store the new values of $a[i]$ in $b[i], i = 1, 2, \ldots, N$.

Step (v): Calculate the value of the objective function for the candidate solution $b[i], i = 1, 2, \ldots, N$, and store it in f_j. Find $del = f_j - f_i$.

Step (vi):

If ($del < thresh$) then
  $f_i = f_j$ and
  $a[i] = b[i]$, for all $i = 1, 2, \ldots, N$.
  $new = f_i$ and go to Step (vii).
else
  $new = f_i$
  $a[i] = -a[i]$
  $i = i + 1$ and go to Step (iv).

Step (vii):

if ($thresh < thrtol$) then
  del $2 = (new - old)/old$
  if(abs(del2)$<acc$) then report $b[i]$, $i = 1, 2, \ldots, N$, as the global optimal solution with f_j as the global optimum.

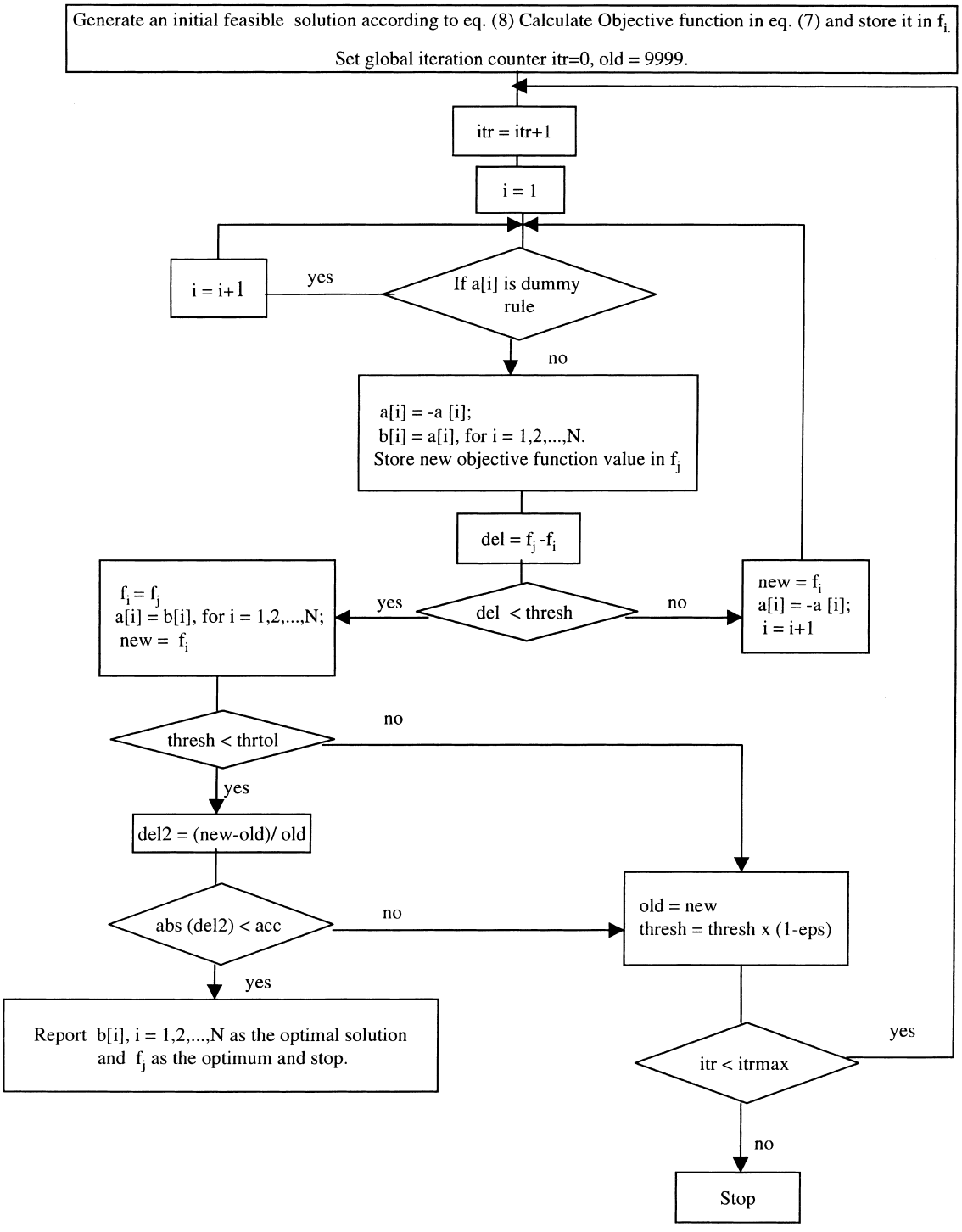

Fig. 2. Flowchart of the MTA algorithm.

Table 1
Results of example 1 (Training data used as test data)

# Partitions	Operator									
	Product		*Minimum*		γ-operator		Fuzzy *and*		*Min/max*	
	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$
5	100	14	99.44	13	100	16	98.88	65	98.88	42
4	98.88	13	98.88	13	98.88	13	100	11	98.31	15
3	98.88	15	97.75	12	98.88	15	95.51	12	97.75	12

Note: $0 \leqslant \gamma \leqslant 1$; *Min/max* indicates a convex combination of *Min* and *Max* operators.

Table 2
Results of example 1 (Leave-one-out technique)

# Partitions	Operator									
	Product		*Minimum*		γ-operator		Fuzzy *and*		*Min/max*	
	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$
5	100	4.71	100	4.71	100	4.71	100	54.3	95.51	52.2
4	100	3	93.26	2.8	93.26	2.8	100	2.96	96.63	2.89
3	99.44	3	99.44	3.36	93.26	3.2	98.87	3.03	98.87	3.31

Notes $0 \leqslant \gamma \leqslant 1$; *Min/max* indicates a convex combination of *Min* and *Max* operators.
In this case, the average number of rules are presented.

Table 3
Results of example 2 (Training data used as test data)

# Partitions	Operator									
	Product		*Minimum*		γ-operator		Fuzzy *and*		*Min/max*	
	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$
8	97.8	26	97.66	29	97.8	25	93.85	3	93.99	12
7	97.8	28	97.51	23	97.8	28	93.7	6	93.56	4
6	97.36	24	96.93	21	97.21	23	93.56	3	93.56	4
5	96.78	20	95.75	18	95.9	14	94.58	6	93.56	6
4	95.46	14	95.75	15	94.29	9	94.58	6	94.14	3
3	95.02	11	94.14	6	94.14	6	94.43	6	93.56	5

Note: $0 \leqslant \gamma \leqslant 1$; *Min/max* indicates a convex combination of *Min* and *Max* operators.

Table 4
Results of example 2 (Leave-one-out technique)

# Partitions	Operator									
	Product		*Minimum*		γ-operator		Fuzzy *and*		*Min/max*	
	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$	C.P	$\|S\|$
8	100	3	99.85	2.99	100	3	87.84	6.12	99.56	3.03
7	100	3	100	3	100	3	93.99	3.75	100	3.93
6	100	3	100	3	100	2.99	93.56	3.74	93.56	3.74
5	98.54	2.95	98.68	2.94	98.54	2.95	93.56	3.74	93.56	3.74
4	98.83	2.96	98.83	2.96	98.54	2.95	98.97	2.97	96.07	2.88
3	97.36	2.92	96.93	2.91	96.78	2.89	88.58	3.54	88.29	3.53

Notes $0 \leqslant \gamma \leqslant 1$; *Min/max* indicates a convex combination of *Min* and *Max* operators.
In this case, the average number of rules are presented.

else
$old = new$;
$thresh = thresh^*$ (1-*eps*)
Step (viii): if ($itr < itrmax$) go to step (iii). □

In the above algorithm, *thresh* is the initial threshold value, *eps* is the factor used in the reduction of the threshold, *thrtol* is the prespecified small value beyond which the threshold is not reduced, *acc* is the prespecified small value used in the convergence criterion, *old* is the arbitrarily specified initial value for the objective function and *itr* is the global iteration counter, *itrmax* is the maximum number of global iterations. The user can change the values of the *eps*, *thrtol* and *acc* which control the speed of convergence of the algorithm and accuracy of the optimal solution. The values of *thresh*, *acc*, *old* and *itrmax* are kept constant at the values specified in step (iii), whereas, the *eps* and *thrtol* are taken as parameters and their values are changed arbitrarily as is done in simulated annealing until no non-dominated solution is obtained anymore.

The inner iterations continue until the status of all the non-dummy rules is changed, one at a time, in the candidate solution. Thus, all these combinations constitute the set of neighbourhood solutions for the candidate solution vector. If the changed status of any rule does not increase the classification power then its status is reverted back to the old position. This ensures that only those rules which increase the classification power are retained for the next iteration and the ineffective rules are dropped.

The modification suggested by the us to the original threshold accepting algorithm lies in the manner in which each neighbourhood solution vector is generated in a deterministic fashion from the given candidate solution vector. It is described in Steps (iv) to (vi) in the above algorithm. This is in contrast to the original threshold accepting algorithm presented in [8], where both the initial solution and the neighbourhood solution are chosen either deterministically or randomly.

Further, another variant of the current MTA algorithm is tested on a few cases which produced good results. We call this variant 'MTA with early exit'. That is, we make an early exit from the inner iterations without performing the swapping operation in Step (vi) on all the non-dummy rules in the candidate solution vector. That means we do not test all the possible neighbourhood solutions of a given candidate solution but only a few of them. This strategy is analogous to the one in Non-equilibrium Simulated Annealing [5], where also the inner iterations are prematurely terminated before attaining 'temperature equilibrium'. This strategy works well according to [5] and the experience of the first author [20].

5. Numerical tests and discussion

The data for well known problems in machine learning area can be freely obtained from the Internet website of University of California at Irvine [18]. The first illustrative example worked out using the methodology presented here is the well-known wine classification problem for which the data and documentation are available in the Internet [9]. In a classification context, this is a well posed problem with 'well behaved' class structures. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents (attributes) found in each of the three types of wines. The class distribution for the 178 patterns is as follows: 59 patterns fall in class 1; 71 patterns fall in class 2 and 48 patterns fall in class 3. All attributes are continuous and there are no missing values. The data was used for comparing various classifiers [1,2].

The second numerical example is also a famous one concerning the determination of the breast cancer in humans from Wisconsin University hospital in Madison, USA. This is also freely available in the Internet [16,17] (available via anonymous ftp from ics.uci.edu in the directory /pub/machine-learning-databases/wisconsin-breast-cancer). Samples arrive periodically as Dr. Wolberg reports his clinical cases. The database therefore reflects this chronological grouping of the data. There are at present 699 cases or samples or patterns available whereas in the past usage, only a dataset of size 369 samples was used. Hence their results cannot be compared with our results. The 9

attributes or features which determine whether a patient is benign or malign are: (i) Clump Thickness, (ii) Uniformity of Cell Size, (iii) Uniformity of Cell Shape, (iv) Marginal Adhesion, (v) Single Epithelial Cell Size, (vi) Bare Nuclei, (vii) Bland Chromatin, (viii) Normal Nucleoli, (ix) Mitoses. There are some missing values in 16 samples and hence all those samples were removed completely from our study. Hence, our study comprises only 683 samples or patterns. The class distribution is as follows: Benign: 458 (65.5%) Malignant: 241 (34.5%). This data has been used earlier by Mangasarian et al. [16,17] and Bennet et al. [3].

In this paper, a software in C has been developed on a Pentium 100 MHz machine under Windows 95 platform using a Microsoft Visual C++ 5.0 compiler to implement the model. The *FeatureSelector* has the flexibility of selecting different feature combinations once the user selects the desired number of significant features. The authors desired to have five significant features for both the numerical examples considered here. However, in the Wisconsin breast cancer problem, the *FeatureSelector* selected a particular combination of 3 features which has the least reclassification error and hence more classification power compared to all the combinations of 5 features.

The algorithm is tested in two ways: (i) using the training data itself as the test data (ii) using the leave-one-out technique in the testing phase. The latter method is preferable as there is the danger of over-fitting in the former method. In each of these methods, all the feature spaces have been divided into a maximum of 5 partitions for the wine classification problem and 8 partitions for the Wisconsin breast cancer problem, respectively. This is done in order to keep the computational complexity to a reasonable level, as we work with 5 features in the wine classification problem and 3 features in the Wisconsin breast cancer example. Further, the study has been conducted for 5 cases each corresponding to different aggregator viz. (1) product operator, (2) *min* operator, (3) γ-operator (compensatory *and*), (4) fuzzy *and*, and (5) a convex combination of *min* and *max* operators, that is, $\lambda\ max + (1-\lambda)\ min$, where $0 \leqslant \lambda \leqslant 1$. The optimal compromise solution obtained in the case of compensatory operators depended on the level of compensation i.e. γ, λ etc. They were chosen such that the compromise solution is nondominated in each case. Their values are not presented in the tables for the sake of brevity.

Results of the wine classification problem (see Table 1) show that the fuzzy *and* yielded the best solution of 100% classification power with just 11 rules when 4 partitions were considered. However, the *product* operator performed consistently well and gave the second best solution of 100% classification power with 14 rules, whereas the γ-operator was closely behind giving 100% classification with 16 rules. *Min* operator produced a maximum of 99.44 classification power with 13 rules, in the case of 5 partitions.

The same example when studied with leave-one-out technique (see Table 2), produced vastly different results. When 5 partitions were considered, the *product*, *min* and the γ operators yielded 100% classification power with 4.71 rules on the average. Although fuzzy *and* also produced 100% classification power, it took large number of rules to achieve this. However, when 4 partitions were used, the best compromise solution (in the table) of 100% classification power with 2.96 rules on the average was provided by fuzzy *and* and the *product* operator yielded the next best solution of 100% classification power and 3 rules on the average. High classification rates for all operators were obtained in the case of 3 partitions too.

Ishibuchi et al. [12] reported a solution of 100% classification power with 8 rules in the wine classification problem. However, sufficient information was not provided in [12], as to whether this solution was obtained in the leave-one-out technique or when the training data is used as test data. Hence, our results were not compared with those of [12].

Results of the Wisconsin breast cancer problem (see Table 3) show that the classification power, in general, increases with the fineness of the grid or number of partitions of the fuzzy subspaces. The γ-operator yielded the best solution of 97.8% classification power with 25 rules when 8 partitions were considered. The *product* operator came closely behind with 97.8% classification power and 26 rules when 8 partitions were considered. *Min* operator turned out to be the best among the other three operators.

When the leave-one-out technique was applied to the same problem (see Table 4), the results are as follows. In the case of 6 partitions, 100% classification power with 3 rules on the average was achieved by the *product* and *min* operators whereas the γ-operator produced 100% classification power with 2.99 rules on the average. In the case of 7 partitions, a solution of 100% classification power with 3 rules on the average was produced uniformly by the *product* and *min* and the γ operators, whereas a solution of 100% classification power with 3.93 rules on the average was produced by the convex combination of *min* and *max* operators. However, in the case of 8 partitions, 100% classification power with 3 rules on the average was achieved by the *product* and the γ-operators whereas *min* operator yielded 99.85% classification power with 2.99 rules on the average. Convex combination of *min* and *max* operators outperformed fuzzy *and* in most of the cases.

It is further noticed that the original threshold accepting algorithm was outperformed by the MTA (algorithm in this paper) in the case of both the problems studied here.

6. Conclusions

This paper highlights the paramount aspect of dimensionality reduction of the feature space in classification problems involving a large number of dimensions. The *FeatureSelector*, a software developed solely for feature extraction has been used as a pre-processor for the first time in fuzzy rule based classification problems. A standard fuzzy rule based classification system is modified and invoked with most important features extracted by the *FeatureSelector*. Further, a MTA has been used for minimizing the number of rules in the classification system while guaranteeing high classification power. The number of rules used and the classification power are taken as the objectives for this multi objective combinatorial optimization problem.

The methodology proposed here has been successfully demonstrated for the well-known wine classification problem, which includes 13 feature variables and the Wisconsin breast cancer problem which has 9 feature variables. The authors desired to select 5 most significant feature variables in both the problems using the software plug-in *FeatureSelector*. Results of both examples demonstrate the predominant influence of the type of aggregator on the classification power of the algorithm. The product operator, the γ-operator and fuzzy *and* provided consistently good solutions achieving 100% classification in leave-one-out testing phase. The number of partitions of the feature spaces had also influenced the compromise solutions in both the problems. In general, it was observed that the finer partitions of the feature spaces led to better classification power accompanied by increased number of rules. The high classification powers obtained for both the problems when working with less feature variables than the original number is the significant achievement of this study and it demonstrates the power of the *FeatureSelector*, the strength of the classification algorithm and the efficacy of the combinatorial optimization algorithm, MTA, used in this paper. Further, the MTA has outperformed the original threshold accepting algorithm in the case of both the problems.

Acknowledgements

The first author wishes to thank Deutscher Akademischer Austauschdienst (DAAD) for providing him the financial support for this work through a research fellowship leading to his Ph.D. degree. The authors also express their gratitude to Dr. W.H. Wolberg (physician) University of Wisconsin Hospitals Madison, Wisconsin, USA and Prof. Olvi Mangasarian (mangasarian@cs. wisc. edu) who are principal donors of the breast cancer database to the internet. Thanks are also due to the anonymous referees whose comments have improved the presentation of the paper.

References

[1] S. Aeberhard, D. Coomans, O. de Vel, Comparison of classifiers in high dimensional settings, Technical report no. 92-02, Department of Computer Science and Depart-

ment of Mathematics and Statistics, James Cook University of North Queensland, 1992.
[2] S. Aeberhard, D. Coomans, O. de Vel, The classification performance of RDA technical report no. 92-01, Department of Computer Science and Department of Mathematics and Statistics, James Cook University of North Queensland, 1992.
[3] K.P. Bennett, O.L. Mangasarian, Robust linear programming discrimination of two linearly inseparable sets, Optimization Methods and Software 1 (1992) 23–34.
[4] P.S. Bradley, O.L. Mangasarian, W.N. Street, Feature selection via mathematical programming, INFORMS Journal on Computing 10 (1998) 209–217.
[5] M.F. Cardoso, R.L. Salcedo, S.F. de Azevedo, Nonequilibrium simulated annealing: A faster approach to combinatorial optimization, Industrial Engineering Chemistry Research 33 (1994) 1908–1918.
[6] C. Chang, Dynamic programming as applied to feature subset selection in a pattern recognition system, IEEE Transactions on Systems, Man and Cybernetics 3 (1973) 166–171.
[7] Data Engine 2.0, The User's Manual, MIT GmBH, Aachen, Germany, 1997.
[8] G. Dueck, T. Scheuer, Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing, Journal of Computational Physics 90 (1990) 161–175.
[9] M. Forina et al., Wine Recognition Database, 1991. Available via anonymous ftp from `ics.uci.edu` in directory `/pub/machine-learning-databases/wine` .
[10] H. Ishibuchi, K. Nozaki, H. Tanaka, Distributed representation of fuzzy rules and its application to pattern classification, Fuzzy Sets and Systems 52 (1992) 21–32.
[11] H. Ishibuchi, K. Nozaki, N. Yamamoto, H. Tanaka, Selecting fuzzy if–then rules for classification problems using genetic algorithms, IEEE Transactions on Fuzzy Systems 3 (1995) 260–270.
[12] H. Ishibuchi, T. Murata, Minimizing the fuzzy rule base and maximizing its performance by a multi-objective genetic algorithm. 6th FUZZ-IEEE, Barcelona, Spain, 1997, pp. 259–264.
[13] H. Ichihashi, T. Shirai, K. Nagasaka, T. Miyoshi, Neuro-Fuzzy ID3: A method of inducing fuzzy decision trees with linear programming for maximizing entropy and an algebraic method for incremental learning, Fuzzy Sets and Systems 84 (1996) 1–19.
[14] J. Kittler, P.C. Young, A new approach to feature selection based on the Karhunen–Loeve expansion, Pattern Recognition 5 (1973) 336–352.
[15] B. Kosko, Neural Networks and Fuzzy Systems, Prentice-Hall., 1992, Englewood Cliffs, NJ.
[16] O.L. Mangasarian, W.H. Wolberg, Cancer diagnosis via linear programming, SIAM News 23 (1990) 1–18.
[17] O.L. Mangasarian, R. Setiono, W.H. Wolberg, Pattern recognition via linear programming: Theory and application to medical diagnosis, in: T.F. Coleman, Y. Li (Eds.), Large-scale numerical optimization, SIAM Publications, Philadelphia, 1990, pp. 22–30.
[18] P.M. Murphy, D.W. Aha, UCI repository of machine learning data bases, Department of Information and Computer Science, University of California, Irvine, (1994) `www.ics.uci.edu./mlearn/ML`.
[19] I.H. Osman, J.P. Kelly, Meta-heuristics: An overview, in: I.H. Osman, J.P. Kelly (Eds.), Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, Boston, MA, 1996, pp. 1–21.
[20] V. Ravi, B.S.N. Murty, P.J. Reddy, Nonequilibrium simulated annealing algorithm applied to reliability optimization of complex systems, IEEE Transactions on Reliability 46 (1997) 233–239.
[21] V. Ravi, P.J. Reddy, H.J. Zimmermann, Fuzzy rule base generation and its minimization via modified threshold accepting, Fuzzy Sets and Systems (forthcoming).
[22] J. Strackeljan, D. Behr, F. Detro, Feature Selector: A plug-in for feature selection with *DataEngine*, 1st International Data Analysis Symposium, Aachen, Germany, 1997.
[23] J. Strackeljan, D. Behr, T. Kocher, Fuzzy pattern recognition for automatic detection of different teeth substances, Fuzzy Sets and Systems 85 (1997) 275–286.
[24] WINROSA, Manual, MIT GmbH, Aachen, Germany, 1997.
[25] Y. Yuan, M.J. Shaw, Induction of fuzzy decision trees, Fuzzy Sets and Systems 69 (1995) 125–139.
[26] Y. Yuan, H. Zhuang, A genetic algorithm for generating fuzzy classification rules, Fuzzy Sets and Systems 84 (1996) 1–19.