Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 168 (2006) 181–199
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Computing, Artificial Intelligence and Information Technology

Mathematical programming based heuristics for improving LP-generated classifiers for the multiclass supervised classification problem

Jan Adem *, Willy Gochet

Department of Applied Economics, Katholieke Universiteit Leuven, Naamsestraat 69, Leuven 3000, Belgium

Received 9 January 2003; accepted 14 April 2004
Available online 2 July 2004

Abstract

Mathematical programming is used as a nonparametric approach to supervised classification. However, mathematical programming formulations that minimize the number of misclassifications on the design dataset suffer from computational difficulties. We present mathematical programming based heuristics for finding classifiers with a small number of misclassifications on the design dataset with multiple classes. The basic idea is to improve an LP-generated classifier with respect to the number of misclassifications on the design dataset. The heuristics are evaluated computationally on both simulated and real world datasets.

Keywords: Heuristics; Mixed integer linear programming; Supervised classification

1. Introduction

Supervised classification (SC) problems have been intensively studied by researchers of all kind. Statisticians, computer researchers, engineers, biologists, etc. have each developed their own ways of solving the SC problems they were up against. As a result, the practitioner is offered a wide range of techniques to solve his particular SC problem such as neural networks, classification trees, nearest neighbor methods, support vector machines, etc. Recent textbooks such as [7,12,13,27] provide excellent overviews of the SC research domain.

* Corresponding author. Tel.: +32 16 32 69 61; fax: +32 16 32 67 32.
E-mail addresses: jan.adem@econ.kuleuven.ac.be (J. Adem), willy.gochet@econ.kuleuven.ac.be (W. Gochet).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2004.04.031

Various attempts to solve supervised classification problems with mathematical programming (MP) techniques in the past gave rise to a rich variety of MP problems. Ref. [8] gives a good overview until 1990. Most of these MP problems are linear programming (LP) problems. Their objective function is to optimise some kind of linear distance measure, which sometimes acts as a surrogate for design dataset error rate minimization. Contrary to LP problems, mixed integer linear programming (MILP) problems can directly attack the objective of minimizing the number of misclassifications on the design dataset. In the past, several researchers [1,10,16,23] have tried to formulate SC problems as MILP problems. While in the 1980's and before, the attention of researchers was mainly directed at exploring different LP and MILP formulations, in the 1990's the focus point shifted towards finding good solution methods to solve these MILP problems since, in contrast to LP problems, they suffer from computational difficulties. The efforts are mainly focussed at solving efficiently the two-group SC problem. For example [2,4,14,15,18–20,22] have developed solution methods for the two-group SC problem, both optimal and heuristic. As far as the optimal solution methods are concerned, it is not clear what can be expected with regard to real world datasets. However, there are a few very good MP based heuristics [6,19] that can solve real world two-group SC problem instances fast. Although there exist ways to solve a multiclass SC problem by means of solving several two-group problems, such approaches bring about new problems [24]. Hence, it may be interesting to look for MP based heuristics that avoid such problems and can tackle the multiclass SC problem directly. We present such an approach. An alternative way to solve the multiclass problem directly, would be to use parallel discriminators [9]. Mathematical theory on design dataset error rate minimization can be found in [26].

The aim of our work is to develop efficient MP based heuristics for SC problems with more than two classes. In Section 2, we describe the SC problem, present a MILP formulation and discuss some properties of its LP relaxation. The optimal solution of the LP relaxation of this MILP formulation will be the starting point of our MP based heuristics. The heuristics will be described in Section 3. Section 4 gives computational results on both simulated and real world datasets. To end, the conclusions are summarized.

2. Mathematical programming formulation

The SC problem consists of finding a formal rule that classifies patterns with unknown class membership into one of a finite number of classes C as accurately as possible. If $C = 2$ the problem is called the two-group SC problem. Such a formal rule is called a classifier. Patterns are whatever needs to be classified. For each pattern the values of P measurements are known. Selection of measurements that are meaningful for class membership prediction is a difficult problem in itself and commonly referred to as feature selection and feature extraction [27]. In order to design a classifier, a design dataset consisting of a finite number of patterns N with known class membership is given. Denote by N_c the number of patterns that belong to class $c \in \{1,\ldots,C\}$. Let $x_{np} \in \mathrm{IR}$ with $n \in \{1,\ldots,N\}$ and $p \in \{1,\ldots,P\}$ be the value of measurement p of pattern n and $c_n \in \{1,\ldots,C\}$ the class to which pattern n belongs. Overviews of the currently available techniques for designing classifiers are given in textbooks such as [27]. One way of construction is through the use of C functions of the form

$$g_c : \mathrm{IR}^P \mapsto \mathrm{IR} : (x_1,\ldots,x_P) \mapsto g_c(x_1,\ldots,x_P) = a_{c0} + \sum_{p=1}^{P} a_{cp} x_{np}$$

with $a_{c0}, a_{cp} \in \mathrm{IR}$ [3,11]. The functions are labelled such that function g_1 is associated with class 1, function g_2 with class 2, etc. This link suggests the following classifier: classify a pattern with measurements $x_1,\ldots,x_P$ into one of the classes c^* for which $c^* = \arg\max_c\{g_c(x_1,\ldots,x_P)\}$. If a classifier classifies a pattern into an incorrect class then the pattern is referred to as a misclassification. Let ϵ be a small strictly positive

constant and M a sufficiently large strictly positive constant. By introducing the N binary variables $y_n = 1$ if pattern n is misclassified and $= 0$ otherwise, the classifier with the minimum number of misclassifications on the design dataset is obtained by solving the following MILP problem:

MILP-ϵM:

$$\min_{a_{cp}, y_n} \quad \sum_{n=1}^{N} y_n$$

$$\text{s.t.} \quad \left(a_{c_n 0} + \sum_{p=1}^{P} a_{c_n p} x_{np}\right) - \left(a_{c0} + \sum_{p=1}^{P} a_{cp} x_{np}\right) + M y_n \geqslant \epsilon,$$

$$n \in \{1, \ldots, N\}, \quad c \in \{1, \ldots, c_n - 1, c_n + 1, \ldots, C\},$$

$$a_{cp} \in \mathrm{IR}, \quad c \in \{1, \ldots, C\}, \ p \in \{0, 1, \ldots, P\},$$

$$y_n \in \{0, 1\}, \quad n \in \{1, \ldots, N\}.$$

The size of MILP-ϵM is a function of N, P and C: the number of constraints in MILP-ϵM is $N(C-1)$ while there are N binary and $C(P+1)$ continuous variables. Without loss of generality, one of the functions g_c can be set to 0 [11]. Setting the right-hand side to ϵ rather than zero prohibits the trivial solution from being optimal, i.e. the solution with $a_{cp} = a_{dp}$ for $c, d \in \{1, \ldots, C\}$ and $p \in \{0,1, \ldots, P\}$. It is easy to check that the value of ϵ only affects the optimal solution up to a multiplicative scalar and does not change the objective function value of MILP-ϵM provided M is sufficiently large. The optimal objective function value of MILP-ϵM is invariant to any linear transformation of the measurements and the optimal solution can be transformed accordingly (see Appendix A). M should be sufficiently large but no value can be given which is guaranteed to be large enough as for any value of M it is possible to construct an MILP-ϵM instance for which it is too small and cuts away the optimal solution (see Appendix B). Such instances are however unlikely to be encountered in practice. As MILP-ϵM always has a nonempty feasible region, a classifier with the minimum number of misclassifications on the design dataset always exists.

The classifier obtained by solving the MILP-ϵM model is more robust to outliers than the classifiers found by LP models that minimize some kind of linear distance measure. Consider the small dataset with three classes in Fig. 1. The first panel displays the classifier that was obtained by solving the MILP-ϵM LP relaxation. It has zero misclassifications. Exactly the same classifier was obtained by solving MILP-ϵM. Next, the dataset was distorted by an outlier and solved again. The MILP-ϵM solution stayed the same, as shown in the second panel. There is only one misclassification (indicated by a hollow symbol). However, the third panel reveals that the MILP-ϵM LP relaxation solution has changed drastically. Note that by

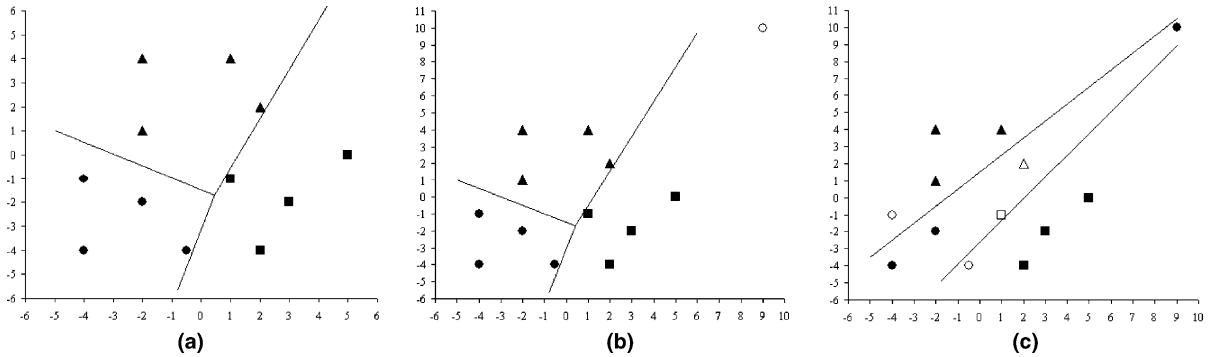

Fig. 1. Robustness to outliers: (a) no outliers; (b) one outlier, MILP-ϵM classifier; (c) one outlier, MILP-ϵM LP-relaxation classifier.

adjusting itself to the outlier, the new MILP-ϵM LP relaxation classifier completely fails to reflect the class structure adequately.

The model can also easily deal with nonlinear classifiers. Consider the 3-class dataset in Fig. 2. It cannot be separated perfectly in the (x_1, x_2) measurement space. The first panel shows the MILP-ϵM classifier. Now let $x'_1 = x_1^2$ and $x'_2 = x_2$ and calculate the MILP-ϵM classifier in the (x'_1, x'_2) measurement space. The resulting classifier has zero misclassifications in the (x'_1, x'_2) measurement space and is shown in panel 2. The third panel shows the corresponding (nonlinear) classifier in the original (x_1, x_2) measurement space. It has zero misclassifications.

An advantage of using the MP approach for solving SC problems is the ability to model more complex SC problems. Assume that for a particular SC problem one knows the cost q_n of misclassifying each pattern. Further, the problem also requires that for class 2, a design dataset error rate of at least 90% should be achieved and that at least two of the patterns 1, 2 and 5 should be classified correctly by the classifier. In approaches such as neural networks, classification trees or nearest neighbor methods, it is far from easy to incorporate such requirements. In the MP approach, all one has to do is change the objective function of MILP-ϵM to $\min_{a_{cp},y_n} \sum_{n=1}^{N} q_n y_n$ and add the constraints $\sum_{n=1,c_n=2}^{N} y_n \leqslant \lfloor N_2(1-0.90) \rfloor$ and $y_1 + y_2 + y_5 \leqslant 1$. The MP formulation will then look for a minimal cost classifier that satisfies all of the demands (if such classifier exists).

The optimal solution of the LP relaxation of MILP-ϵM is the starting point of the MP based heuristics. The LP relaxation of MILP-ϵM can be written as

$$\min_{a_{cp}} \left\{ \sum_{n=1}^{N} \left(\max_{c} \left\{ \frac{\left(a_{c0} + \sum_{p=1}^{P} a_{cp} x_{np} \right) - \left(a_{c_n 0} + \sum_{p=1}^{P} a_{c_n p} x_{np} \right) + \epsilon}{M}, 0 \right\} \right) \right\}$$

with $a_{cp} \in \mathrm{IR}$. The values of M and ϵ only scale the objective function value of the LP relaxation. The optimal solution is the same for all strictly positive values M and ϵ. The strictly positive value of ϵ does not prevent the trivial solution from being optimal in the LP relaxation. As in [3] sufficient and necessary conditions for the trivial solution to be optimal in the LP relaxation can be derived (see Appendix C) and are independent of M and ϵ. The optimal objective function value of the MILP-ϵM LP relaxation is also invariant to any linear transformation of the measurements and the optimal solution can be transformed accordingly. The feasible region of the LP relaxation is always nonempty. The LP relaxation lower bound is weak in general. The gap will only be zero when a classifier with zero misclassifications on the design dataset exists.

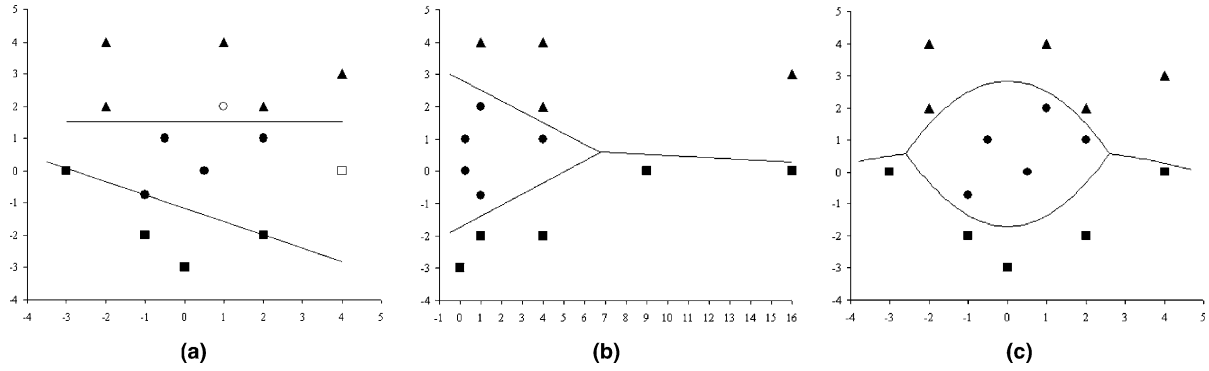

Fig. 2. Finding nonlinear classifiers: (a) classifier 1, (b) classifier 2, (c) (nonlinear) classifier 2.

Note that the objective function of both the MILP-ϵM and its LP-relaxation are only determined by the patterns that are misclassified, or, correctly classified but less than ϵ away from the decision boundary. In the SVM approach [13], a similar phenomenon is observed and such patterns are called support vectors.

3. Mathematical programming based heuristics

Solving MILP-ϵ to optimality is computationally difficult in general. LINDO quickly runs into extremely large computation times or software problems. We have tried to write a more efficient branch and bound algorithm using different branching and search strategies in the branch and bound tree. Unfortunately, there was little consistency in the computational results: which strategy works best is highly dependent on the design dataset to be solved and, in general, the computational effort needed remains high. The fundamental problem with the branch and bound idea is the lack of a good lower bound. Similar research for the two-group SC problem points to the same conclusion [20].

We have developed two sets of heuristics. The common idea behind both sets of heuristics is to improve the LP-generated classifier with respect to the number of misclassifications on the design dataset.

3.1. AG heuristics

Sometimes it is not difficult to improve an LP-generated classifier with respect to the number of misclassifications on the design dataset. Consider the stage 1 classifier in Fig. 3(a) obtained by solving the MILP-ϵM LP relaxation.

The stage 1 classifier has three misclassifications on the design dataset (indicated by hollow objects in Fig. 3(a)). The full lines indicate the classification boundary while the dashed lines show the boundaries $+\epsilon$ and $-\epsilon$. For all patterns n that are correctly classified, add the constraint $y_n = 0$ to the MILP-ϵM LP relaxation formulation. The addition of these constraints ensures that any new optimal MILP-ϵM LP relaxation solution will keep these patterns (indicated by nonhollow objects in Fig. 3(a)) correctly classified. Choose one of the patterns that are incorrectly classified, say pattern $\dot{n}$ with measurements $(-1, -1)$ and add the constraint $y_{\dot{n}} = 0$ to the MILP-ϵM LP relaxation formulation and solve it. Fig. 3(b) displays the new optimal classifier, labelled stage 2 classifier. It has only two misclassifications. Again choose one of the patterns that are incorrectly classified, e.g. pattern $\dot{n}$ with measurements $(1, 1)$. Add the constraint

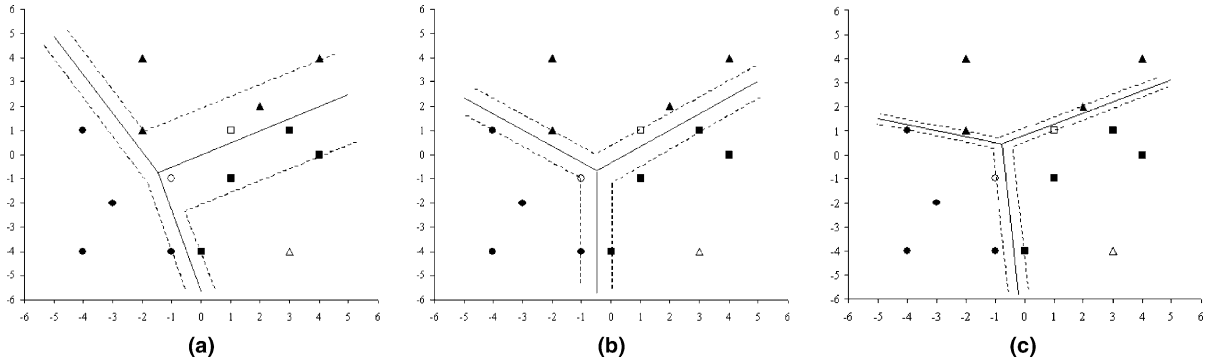

Fig. 3. Improving the MILP-ϵM LP relaxation classifier: (a) stage 1 classifier, (b) stage 2 classifier, (c) stage 3 classifier.

$y_{\dot{n}} = 0$ and solve the MILP-ϵM LP relaxation formulation which yields the stage 3 classifier shown in Fig. 3(c). The stage 3 classifier is an optimal solution to the MILP-ϵM instance.

Iteratively applying this idea leads to a heuristic for improving LP-generated classifiers with respect to the number of misclassifications on the design dataset. A formalized description of the heuristic is given below.

Denote by $\mathbf{X} \in \mathrm{IR}^{N\times P}$ the matrix of measurements of the design dataset patterns. Let $(a^*_{c0}, \mathbf{a}^*_c, \mathbf{y}^*)$ with $\mathbf{a}^*_c = [a^*_{c1} \cdots a^*_{cP}]$ and $\mathbf{y}^* = [y^*_1 \cdots y^*_N]$ with $c \in \{1,\ldots,C\}$ and $n \in \{1,\ldots,N\}$ represent the optimal solution of the MILP-ϵM LP relaxation.

input: N, C, P, $\mathbf{X}$
initialize: $ub_{best} = N$
$solution_{best} = (0,\mathbf{0},\mathbf{0})$
$level = 0$
solve MILP-ϵM LP relaxation
step 1: $level \leftarrow level + 1$
$solution_{current} = (a^*_{c0}, \mathbf{a}^*_c, \mathbf{y}^*)$
$F_{level} = \{n | c_n \neq \arg\max_c \{g_c(x_{n_1},\ldots,x_{nP})\}\}$
$ub_{current} = |F_{level}|$
if $ub_{current} < ub_{best}$ then
$ub_{best} \leftarrow ub_{current}$
$solution_{best} \leftarrow solution_{current}$
goto step 2
step 2: if $|F_{level}| = 0$, goto step 6, else goto step 3.
step 3: choose $\dot{n} \in F_{level}$
change $F_{level} \leftarrow F_{level} \setminus \{\dot{n}\}$
add constraints $y_n = 0$ for $n \notin F_{level}$
add constraint $y_{\dot{n}} = 0$
solve MILP-ϵM LP relaxation
goto step 4
step 4: if feasible, goto step 1, else goto step 5
step 5: remove constraint $y_{\dot{n}} = 0$, goto step 2
step 6: stop
output: ub_{best}
$solution_{best}$

Only in the initialization, it is possible to obtain the trivial optimal solution. This solution sets $ub_{current} = N$ but does not jeopardize the correct functioning of the heuristic. There are several possible strategies for choosing a pattern $\dot{n} \in F_{level}$ in step 3. These strategies are expected to determine the computational efforts of the heuristic and the quality of the upper bound. A strategy that is guaranteed to find the MILP-ϵM optimum cannot be found.

AG-y Select a pattern $\dot{n} = \arg\min_n \{y_n | n \in F_{level}\}$. A small y_n value indicates that pattern n is close to being classified correctly. Given the correctly classified patterns, it should be relatively easy to classify this pattern also correctly.

AG-stLP Calculate for each pattern $\tilde{n} \in F_{level}$ the MILP-ϵM LP optimum $z_{\tilde{n}}$ after adding the constraints $y_n = 0$ for $n \notin F_{level}$ and $y_{\tilde{n}} = 0$. Select a pattern $\dot{n} = \arg\min_{\tilde{n}} \{z_{\tilde{n}} | \tilde{n} \in F_{level}\}$. The number of elements in F_{level} determines the number of extra LP problems to be solved and strongly influences the computation time.

AG-stIP The same as in AG-stLP but now based on $ub_{\tilde{n}}$ values in stead of $z_{\tilde{n}}$ where $ub_{\tilde{n}}$ is the value of the upper bound associated with the optimal MILP-ϵM LP solution.

AG-stIPs Calculate for each pattern $\tilde{n}$ in F_{level} the set $S_{\tilde{n}} = \{n | y_n = 0\}$ and the upper bound $ub_{\tilde{n}}$ in the MILP-ϵM LP optimum after adding the constraints $y_n = 0$ for $n \notin F_{level}$ and $y_{\tilde{n}} = 0$. Select those patterns $\tilde{n}$ for which $S_{\tilde{n}}$ is no subset of any other set $S_{\breve{n}}$ with $\breve{n} \in F_{level}$ and $\breve{n} \neq \tilde{n}$. Backtracking is done only on these patterns $\tilde{n}$. As in AG-stIP, we first select a pattern $\dot{n} = \arg\min_{\tilde{n}}\{ub_{\tilde{n}} | \tilde{n} \in F_{level}\}$.

Backtracking to the previous *level* can be allowed for by changing step 6 of the AG heuristic.

step 6: remove all constraints $y_n = 0$
$level \leftarrow level - 1$
if $level = 0$, then stop, else goto step 2.

Even with backtracking, the heuristic is not guaranteed to find the MILP-ϵM optimum. Backtracking on all elements of F_{level} is to be avoided since it might be very time consuming (see Section 4). We choose to backtrack only on the k most promising elements of F_{level}, if there exist k elements. The procedure above corresponds to $k = 1$. We have implemented $k = 2$ and $k = 3$. A strategy for choosing $\dot{n} \in F_{level}$ only makes sense if we do not backtrack on all elements of F_{level}.

In the course of the algorithm, we continuously add and remove constraints to the MILP-ϵM LP relaxation. In general, it is more efficient to solve the dual of the MILP-ϵM LP relaxation since it has a better computational structure than the primal. Adding or removing primal constraints of the type $y_n = 0$ comes down to adding and removing free variables with the dual objective function coefficient 0 in the dual. The cost of updating the optimal LP solution in the dual after such addition or removal is relatively small because reoptimisation starts from the previous optimal solution which usually gives a good starting basis such that only a small number of simplex iterations are needed.

3.2. CH heuristics

Chinneck [6] developed polynomial time heuristics for the maximum feasible subsystem (MAX FS) problem. The idea of the heuristics is to iteratively eliminate constraints from the infeasible LP problem until a feasible LP problem is obtained. The number of remaining constraints in this feasible LP gives an upper bound for the MAX FS problem. The heuristics differ in the way they determine which constraint is to be removed. The removal is permanent. Backtracking on the constraint removal is not done which makes the procedures relatively fast but heuristic. A formal description of the heuristics can be found in [6]. The heuristics can be readily applied to the two-group SC problem. The empirical results in [6] indicate that the classifiers have a high accuracy on the design dataset and are found fast.

It is straightforward to extend these heuristics to SC problems with more than two classes. Observe that in the two-group SC problem, for each pattern exactly one constraint is in the MILP-ϵM formulation and that the y_n variables in the MILP-ϵM LP relaxation have the same function as the elastic variables in the elasticised version of the infeasible LP in the MAX FS problem. Iteratively removing constraints from the LP relaxation then comes down to iteratively removing patterns. The same idea, iteratively removing patterns, can also be used for the SC problem with more than two classes. Removing one pattern corresponds to deleting $C - 1$ constraints. The function of the y_n variables in the MILP-ϵM LP relaxation remains unchanged, though now each y_n corresponds to $C - 1$ constraints. Each time the MILP-ϵM LP relaxation is solved, the corresponding MILP-ϵM upper bound is calculated (see step 1 of the AG heuristic) and the

classifier with the lowest upper bound value is saved as the final solution. This upper bound value might be smaller than the number of patterns that has to be removed in order to have a MILP-ϵM LP relaxation with perfect separation as nothing guarantees that once a pattern is removed, it will always be misclassified.

Several strategies are possible for deciding on which pattern is to be removed. This step of the procedure is expected to determine the speed and the quality of the upper bound. Again, one cannot find a strategy that is guaranteed to find the MILP-ϵM optimum.

CH-y Select a pattern $\dot{n} = \arg\max_n\{y_n\}$. A large y_n value indicates that pattern n is far from being classified correctly. In this strategy, patterns that are difficult to classify correctly are removed first.

CH-$dp\cdot y$ Select a pattern $\dot{n} = \arg\max_n\{dp_{ny_n}\}$ where dp_n is the largest dual price of the $C-1$ constraints associated with pattern n. dp_{ny_n} is a good estimator for the reduction in the MILP-ϵM LP objective function when the $C-1$ constraints are left out of the formulation and we choose the pattern n that gives the strongest estimated reduction.

CH-stLP Make a set of candidate patterns for removal S_R. Calculate for each pattern $n \in S_R$ the MILP-ϵM LP optimum z_n with that pattern removed. Select a pattern $\dot{n} = \arg\min_n\{z_n \mid n \in S_R\}$. This strategy looks for the pattern n that gives the steepest descent in the MILP-ϵM LP optimum value. The number of elements in the set S_R determines the number of extra LPs to be solved and determines the computation time. As in [6], there are several strategies to build the set S_R. Set $S = \{n \mid y_n > 0\}$. We have implemented the following choices for S_R.

CH-stLP(2): S_R = the elements of S with the 2 largest y_n values
CH-stLP(3): S_R = the elements of S with the 3 largest y_n values
CH-stLP(all): $S_R = S$

CH-stIP The same as in CH-stLP but now based on ub_n values instead of z_n where ub_n is the value of the upper bound associated with the MILP-ϵM LP relaxation optimum.

Again, it is computationally more efficient to work on the dual. Removing the $C-1$ constraints of pattern n in the primal is equivalent to changing the right-hand side coefficient of these constraints to $-M$. Hence, in the dual, we only have to set the $C-1$ objective function coefficients of the variables corresponding to these $C-1$ constraints to $-M$. Relatively few simplex iterations are required to reoptimise this dual LP since the previous optimal solution usually gives a good starting basis.

4. Computational results

The heuristics are tested on both simulated and real world datasets. The aim of the tests on the simulated design datasets is to compare the heuristics for different values of N, P and C. It is not our intention yet to study generalization performance. With respect to the simulated datasets, the only questions of interest are: does the heuristic find a solution close to the global optimum? and how fast does it find this solution? Our best performing heuristics will then be selected and ran on real world datasets to compare their performance to the performance of existing parametric and nonparametric methods.

4.1. Simulated design datasets

For each triplet (N, P, C) with $N \in \{25, 50, 75, 100, 250, 500, 750, 1000\}$, $P \in \{2, 3, 4, 5, 10\}$ and $C \in \{2, 3, 4, 5\}$, 10 design datasets were simulated. Patterns in each of the 1600 simulated design datasets

are drawn from one of C multivariate normal distributions with covariance matrices equal to the identity matrix and the C mean vectors randomly drawn from the space $\{0,1,2,3,4,5,6\}^P$. This space allows for a sufficient amount of variation in the overlap between the different classes in one design dataset. To generate a pattern, a random number i from $\{1,\ldots,C\}$ is drawn and values $x_1, x_2, \ldots, x_P$ are taken from the associated multivariate normal distribution. Having done this N times, it is checked if each class is represented, i.e. $N_c > 0$ for $c \in \{1,\ldots,C\}$. If this is not the case, the N patterns are rejected and resampled. Otherwise, the design dataset is accepted.

The MP based heuristics are implemented in C and make use of the LINDO solver [21]. The experiment was run on a PENTIUM III 550 MHz computer. The LP-generated classifier is the starting solution for all procedures and took on average 0.4 seconds to determine. The 1600 design datasets are divided into four groups as their LP relaxation upper bound falls in $]0,10]$, $]10,50]$, $]50,100]$ or $]100,1000]$. These intervals represent the difficulty of the design dataset instances. 667 design datasets are perfectly separable.

Table 1 gives statistics on the improvement on the LP-generated classifiers achieved by the different procedures. *tot* is the total number of misclassifications of the MP based heuristic solutions on all 1600

Table 1
Results on simulated datasets

Heuristic	*tot* (933)	*tot*]0, 10] (246)	*tot*]10, 50] (332)	*tot*]50, 100] (101)	*tot*]100, 1000] (254)
LP relaxation	100% (40) 00:00:01	100% (40) 00:00:01	100% (0) 00:00:01	100% (0) 00:00:01	100% (0) 00:00:02
AG-y(1)	49.1% (454) 00:00:03	41.4% (233) 00:00:01	41.7% (185) 00:00:01	46.0% (17) 00:00:03	50.5% (19) 00:00:16
AG-stLP(1)	48.8% (469) 00:01:19	41.1% (234) 00:00:01	41.3% (188) 00:00:04	45.3% (26) 00:00:29	50.2% (21) 00:06:57
AG-stIP(1)	51.0% (385) 00:00:25	41.7% (228) 00:00:01	43.5% (143) 00:00:02	48.1% (8) 00:00:09	52.3% (6) 00:02:34
AG-stIPs(1)	50.4% (418) 00:00:35	41.7% (227) 00:00:04	42.3% (171) 00:00:06	47.3% (13) 00:00:19	51.8% (7) 00:03:10
AG-stIPs(2)	48.8% (546) 00:00:41	40.3% (243) 00:00:04	40.2% (250) 00:00:07	45.1% (29) 00:00:26	50.4% (24) 00:03:43
AG-stIPs(3)	48.1% (621) 00:00:53	**40.1**% (246) 00:00:04	39.5% (282) 00:00:10	44.2% (46) 00:00:42	49.8% (47) 00:04:53
AG-stIPs(all)	– –	**40.1**% (246) 00:00:04	**38.8**% (321) 00:09:00	– –	– –
CH-y	47.0% (642) 00:00:25	41.9% (226) 00:00:01	40.7% (209) 00:00:01	43.7% (55) 00:00:09	48.2% (152) 00:02:33
CH-$dp \cdot y$	47.0% (641) 00:00:25	42.1% (224) 00:00:01	40.8% (209) 00:00:01	43.7% (54) 00:00:09	48.2% (154) 00:02:31
CH-stLP(2)	**46.5**% (697) 00:02:37	40.8% (238) 00:00:01	40.2% (234) 00:00:09	43.6% (61) 00:01:44	**47.6**% (164) 00:15:37
CH-stLP(3)	– –	40.4% (242) 00:00:01	40.0% (242) 00:00:18	**43.5**% (68) 00:03:14	– –
CH-stLP(all)	– –	40.2% (245) 00:00:03	39.8% (256) 00:07:41	– –	– –
CH-stIP(2)	46.6% (653) 00:02:40	41.3% (233) 00:00:01	40.5% (217) 00:00:08	43.8% (52) 00:01:40	47.7% (151) 00:15:54
CH-stIP(3)	– –	40.7% (239) 00:00:01	40.4% (216) 00:00:18	43.8% (53) 00:03:46	– –
CH-stIP(all)	– –	41.2% (234) 00:00:03	43.1% (125) 00:08:38	– –	– –

simulated design datasets expressed as a percentage of the number of misclassifications of the LP relaxation solutions. *tot*]*i*, *j*] gives similar percentages but only for the simulated design datasets with an LP relaxation upper bound in]*i*, *j*]. Between brackets is the number of times the procedure yielded the lowest upper bound out of the total number of instances in this group, which is given between brackets in the top row. The average CPU time is given in hh:mm:ss format. A dash (–) indicates that we interrupted the heuristic because of extremely large CPU times on some of the design datasets and hence could not calculate the corresponding statistic. The best improvement for each group of instances is printed in bold.

All the heuristics are able to improve the LP relaxation starting solution considerably for all difficulties of the instances. Surprisingly, search strategies do not matter much. The upper bounds obtained by the more elaborated search strategies (AG-stLP, AG-stIP, AG-stLPs, CH-stLP, CH-stIP) are not significantly better than the upper bounds obtained by the simpler ones (AG-y, CH-y, CH-$dp \cdot y$), though they are definitely more time consuming to calculate. Based on the number of times each procedure yields the lowest upper bound, the CH heuristics outperform the AG heuristics especially if the design dataset gets more difficult to solve. CPU time goes up quickly if the design dataset gets more difficult. AG-y(1) is the fastest heuristic. Note that almost all the CPU time in the iterations of each heuristic (e.g. for AG-y(1) 99%) is absorbed by the LP solver, here the simplex solver of LINDO. For the AG-y, AG-stLP and AG-stIP heuristics, backtracking does not result in large improvements in the upper bounds. The upper bound found after backtracking cannot be worse but computational results on the easy instances in Table 2 indicate that on average the upper bound improvement is small and obtained at the cost of a large increase in CPU time. Additional tests on more difficult instances confirm this conclusion. Therefore, we will not consider the heuristics AG-y(2), AG-y(3), AG-y(all), AG-stLP(2), AG-stLP(3), AG-stLP(all), AG-stIP(2), AG-stIP(3) and AG-stIP(all) further.

An interesting question is how many times we hit the global optimum with each heuristic. 667 design datasets are perfectly separable and LINDO was only able to solve 147 of the remaining 933 simulated datasets to optimality. The maximal pivot limit was hereby set to 1,000,000, allowing for a substantial amount of CPU time. Table 3 gives the number of times the MILP-ϵM optimum was found by the MP-based heuristic out of the number of times LINDO found the MILP-ϵM optimum. For 58 design datasets the MILP-ϵM optimum was 1, for 25 design datasets it was 2, etc.

Search strategy does not seem to have a large impact on the ability to hit the MILP-ϵM optimum. On the easy design datasets, all heuristics obtain the MILP-ϵM optimum a large number of times. On the difficult instances, the results vary. Again the CH heuristics do slightly better than the AG heuristics. It would be interesting to see if this still holds for the difficult designs datasets (MILP-ϵM optimum $\gg$ 10).

In what follows, we will only consider the best representative of each family of heuristics. We opted for the AG-y(1) and CH-y heuristic as they are very fast, use a simple but logical search strategy and obtain

Table 2
Results of backtracking for easy instances

Heuristic	*tot*]0, 10]	Heuristic	*tot*]0, 10]	Heuristic	*tot*]0, 10]
AG-y(1)	41.4%	AG-stLP(1)	41.1%	AG-stIP(1)	41.7%
	00:00:01		00:00:01		00:00:01
AG-y(2)	40.2%	AG-stLP(2)	40.8%	AG-stIP(2)	40.4%
	00:00:01		00:00:01		00:00:01
AG-y(3)	40.2%	AG-stLP(3)	40.7%	AG-stIP(3)	40.2%
	00:00:02		00:00:09		00:00:01
AG-y(all)	40.2%	AG-stLP(all)	40.6%	AG-stIP(all)	40.2%
	00:00:06		00:07:11		00:02:36

Table 3
Number of times each MP based heuristic hits the global optimum

	1	2	3	4	5	6	7	8	9	>9	*tot*
MILP	58	25	10	15	12	6	6	5	3	7	147
LP relaxation	27	1	0	0	0	0	0	0	0	0	28
AG-*y*(1)	57	23	7	13	8	4	3	5	0	4	120
AG-stLP(1)	57	22	9	12	8	3	3	5	1	4	124
AG-stIP(1)	58	21	8	9	8	3	2	2	1	1	113
AG-stIPs(1)	58	21	8	11	8	5	3	2	1	3	120
AG-stIPs(2)	58	24	10	14	9	6	4	4	1	6	136
AG-stIPs(3)	58	24	10	15	10	6	4	5	2	6	140
AG-stIPs(all)	58	24	10	15	10	6	5	5	2	7	142
CH-*y*	56	20	7	13	9	3	3	4	1	6	122
CH-$dp \cdot y$	56	20	6	12	9	3	3	4	1	6	120
CH-stLP(2)	56	23	10	14	10	4	3	5	1	6	132
CH-stLP(3)	57	23	10	15	11	4	3	4	1	6	134
CH-stLP(all)	58	23	10	15	11	5	3	4	1	7	137
CH-stIP(2)	56	21	8	14	9	4	3	5	2	5	127
CH-stIP(3)	57	22	8	15	10	4	3	5	1	6	131
CH-stIP(all)	57	22	9	10	8	5	3	1	0	1	116

good error rates. Only these two heuristics will be tested on the real world datasets for comparison with other parametric and nonparametric methods.

4.2. Real world datasets

In this section, we will try to improve LP-generated classifiers on real world datasets, obtained from the UCI repository at http://kdd.ics.uci.edu [5]. The datasets are publicly available, come from different contexts and are described below.

- Teaching Assistant Evaluation (*tae*). The dataset contains five measurements of evaluations of teaching performance of 151 teaching assistants. There are three roughly equal-sized classes.
- Statlog Heart Disease (*hea*). In this medical dataset, one has to predict whether or not a patient suffers from a heart disease (yes or no) given the results of various medical tests.
- Glass Identification (*gla*). This dataset concerns the identification of six different types of glass based on 10 continuously valued composite measurements.
- BUPA Liver Disorders (*bld*). The dataset contains data on liver disorders that might arise from excessive alcohol consumption for single male individuals. There are two classes, 345 patterns and six measurements.
- Johns Hopkins University Ionosphere (*ion*). This dataset contains radar data. 126 of the 351 radar returns are "good" in the sense that they show evidence of some type of structure in the ionosphere, the others are "bad". There are 34 continuously valued measurements.
- Tic-Tac-Toe Endgame (*ttt*). This database encodes a set of 958 tic-tac-toe endgame configurations. There are nine measurements each of which can take three values.
- Boston Housing (*hou*). This dataset concerns housing values in suburbs of Boston. There are 506 patterns, twelve continuous and one binary-valued attributes. The class variable is constructed in the same way as in [17].

- Vehicle Silhouettes (*veh*). Given are 18 measurements extracted from car silhouettes taken from different angles. The problem is to find out which of four car types is considered. There are 946 patterns and the four classes are (roughly) equal-sized.
- Contraceptive Method Choice (*cmc*). The problem is to predict the contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics. There are 1473 patterns and nine measurements.

The first subsection presents three typical error rate patterns and comments on the related issue of overfitting. In the second subsection, the performance of the classifiers obtained by means of the MP based heuristics is compared to that of classifiers obtained by other approaches for solving SC problems.

4.2.1. Generalization

Often, but not always, the LP-generated classifiers generalize acceptably well. If we now improve them with respect to the number of misclassifications on the design dataset, does their generalization performance improve too? Three typical error rate patterns are observed.

Consider the AG-y heuristic and the *tae*, *hea* and *gla* datasets. Each dataset was randomly split into a design dataset and a testing dataset of (almost) equal size. At each iteration of the AG-y heuristic, the best found solution is taken and both its design and testing dataset error rate are calculated. By doing so, we can see the evolution of both the design (full line) and testing error rate (dashed line) as the algorithm proceeds.

In the first panel of Fig. 4, the testing error rate consistently goes down with the design error rate, despite an overfit. Note that the LP-generated classifier performs poorly. This is the kind of situation in which it pays off to improve the LP-generated classifier, even to the extreme. The middle panel shows the typical situation in which it is possible to improve the LP-generated classifier, but not to the extreme as the testing dataset error rate eventually starts to go up again. The third panel illustrates the case where it is not a good idea to improve the LP-generated classifier.

Further tests on other datasets or with other MP-based heuristics typically yield one of the above three error rate patterns. Further illustrations are given in Fig. 5. The full line shows the evolution of the MILP-ϵM objective function value and the dotted line indicates the 10-fold cross-validation estimate of the error rate on unseen patterns.

Note that overfitting occurs frequently, but not always. Often, only in the last iterations, we start to overfit. In itself, this need not be problematic as it is not our goal to avoid overfitting but rather to obtain good generalization. A problem arises if during the last iterations, the error rate on unseen data also gets worse. Practically, this means that one has to be careful to push error rate minimization to the extreme. Vapnik [26] has developed related theoretical arguments which warrant care when minimizing design data-

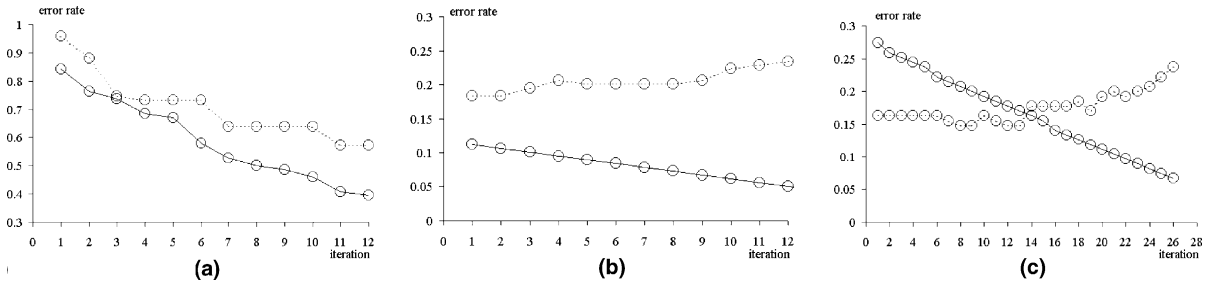

Fig. 4. Generalization performance patterns: (a) *tae*, (b) *hea*, (c) *gla*.

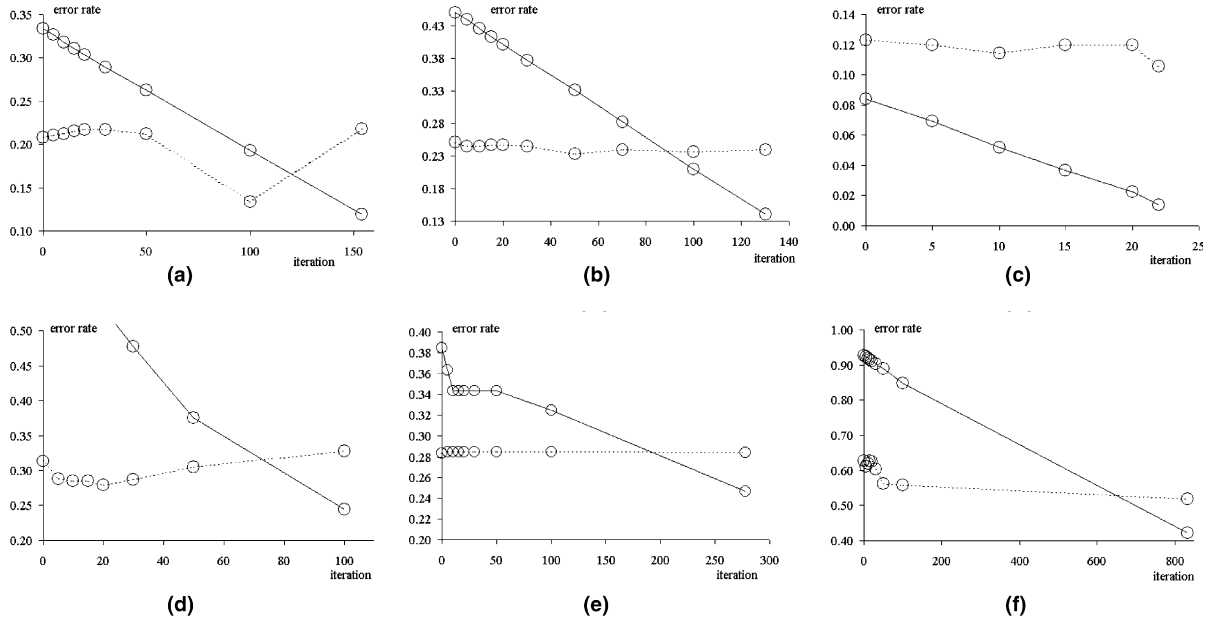

Fig. 5. More generalization performance patterns: (a) *veh* (CH-*y*), (b) *ion* (CH-*y*), (c) *hou* (CH-*y*), (d) *bld* (CH-*y*), (e) *cmc* (CH-*y*), (f) *ttt* (CH-*y*).

set error rate. Therefore, in the following subsection, we will regard the number of allowable iterations as a controllable parameter.

4.2.2. Performance evaluation

Six of the above UCI repository datasets are also used by [17] to test the performance of 33 procedures for SC. Their 10-fold cross-validation procedure was followed in order to allow for comparison of their results with ours. Since they did not give the subsets obtained by stratified sampling on the UCI datasets, we probably did not run our procedures on the exact same subsets. Although efforts are made to level the different measures of CPU time (see Appendix D), the CPU time comparison should be seen as indicative at best.

In Fig. 6, scatter plots of the natural logarithm of the calculation time expressed in seconds versus the cross-validation error rate estimate are presented. The results for the statistical procedures (indicated by ○), the results for the decision trees (indicated by □) and those for the neural networks (indicated by △) can be found in [17]. Our procedures are indicated by ◇'s. The LDA classifier is indicated by a filled black circle. The AG-*y* and CH-*y* procedures were ran for different values of allowable iterations and the best result is shown.

The results indicate that our approach is competitive. Methods that are in the lower left corner of the plot are preferred as they are both fast and obtain a low error rate. On the *hea*, *bld* and *hou* datasets, our methods are among those most in the lower left corner. On the *tae* dataset, the MP based heuristics are fast but the error rate they obtain is only average. The reverse is true for the *veh* dataset where a good error rate is combined with average speed. The performance on the *cmc* dataset is not so good, although the LP relaxation classifier has been improved substantially. Apparently, it is not a good idea to try to separate these data by means of linear functions.

It is interesting to compare the performance of our classifiers to that of the LDA classifiers as they use the same mathematical structure to separate the classes as our classifiers. Note that on three datasets (*tae*,

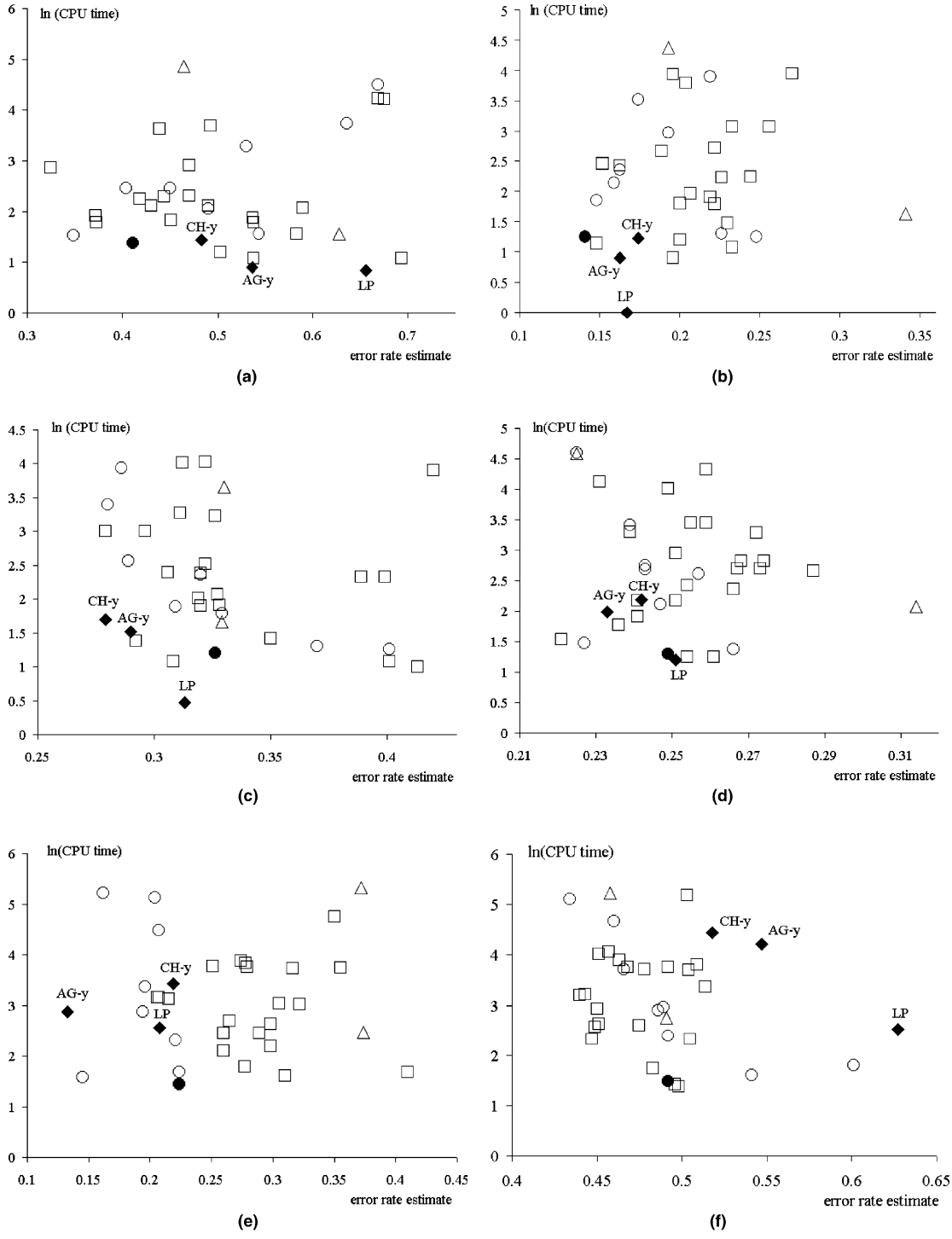

Fig. 6. Results on real world datasets: (a) *tae*, (b) *hea*, (c) *bld*, (d) *hou*, (e) *veh*, (f) *cmc*.

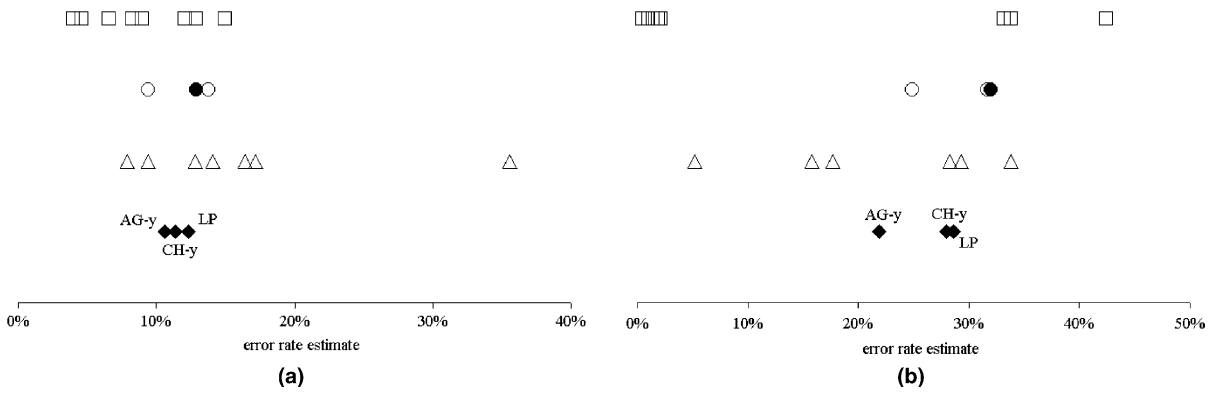

Fig. 7. Results on real world datasets: (a) *ion*, (b) *ttt*.

hea and *cmc*), the LDA classifier does better than our classifiers. These three datasets have relatively many categorical measurements, while in the *bld* and *veh* datasets, all the measurements are continuous and in the *hou* dataset, there is only one out of 13 measurements categorical. Hence, given many categorical variables, it seems advisable to use LDA rather than our approach.

The *ion* and *ttt* dataset are known to have a nonlinear character. As to these two datasets, our results can be compared to those of [25] although they used a slightly different method to estimate the error rate. They tested eighteen procedures on several UCI datasets. Unfortunately, they do not report calculation times. In Fig. 7(a) and (b), plots are constructed using the symbol ○ for statistical procedures, □ for LS-SVM methods and △ for the other methods used in [25]. The other procedures are indicated as in Fig. 6.

On both datasets, the nonlinear classifiers produce the lowest error rate estimates. This confirms the nonlinear character of the datasets. The performance of our procedures is average. On both datasets, the LP relaxation classifier has been improved, though only slightly.

No approach for SC is superior to all other approaches on all dataset settings. If one approach outperforms another approach on a particular dataset setting, it is a consequence of its fit to the particular instance, and not of the superiority of the approach [7]. The fit then is determined by many elements such as dimension of the measurement space, structure of the classifier, sample size, properties of the representation language, search strategies, etc. As for any other technique, it would be interesting to know on what type of SC problems the MP approach works well. This is an essentially empirical question and is a topic of future research.

5. Conclusions

Mathematical programming formulations that minimize the number of misclassifications on the design dataset are hard to solve to optimality. The mathematical programming based heuristics we have brought forward present an alternative to the optimal solution methods. Computational experiments show that they are fast and obtain solutions often close to the global optimum. Experiments on real world datasets indicate that improving the LP-generated classifier on the design dataset often yields improvements on the testing dataset too. In terms of error rate and speed, our approach is competitive to other approaches. The advantage of the mathematical programming approach in the context of supervised classification lies in its the power to model more complex real world supervised classification problems. Exploring such possibilities is part of future research.

Acknowledgement

The authors would like to thank the three anonymous referees for their valuable suggestions.

Appendix A

Denote by $\mathbf{X}_c \in \mathrm{IR}^{N_c \times P}$ the matrix of measurements of patterns from class c. Let $\mathbf{U} \in \mathrm{IR}^{P \times P}$ be a nonsingular matrix and $\mathbf{u} \in \mathrm{IR}^{1 \times P}$ a vector. $\mathbf{e}$ is a column vector of ones of appropriate dimension. Denote by MILP$^{\mathrm{tr}}$-ϵM the MILP-ϵM formulation with measurement vectors $\mathbf{X}_c^{\mathrm{tr}} = \mathbf{X}_c\mathbf{U} + \mathbf{eu}$. The optimal solution to MILP-ϵM is given by $(a_{c0}^*, \mathbf{a}_c^*)$ with $\mathbf{a}_c^* = [a_{c1}^* \cdots a_{cP}^*]^{\mathrm{T}}$ and $c \in \{1, \ldots, C\}$ with objective function value z^*. Then, an optimal solution to MILP$^{\mathrm{tr}}$-ϵM is $(a_{c0}^* - \mathbf{u}\mathbf{U}^{-1}\mathbf{a}_c^*, \mathbf{U}^{-1}\mathbf{a}_c^*)$ with $c \in \{1, \ldots, C\}$ with the same objective function value z^*.

Proof. Assume there exists a solution $(\tilde{a}_{c0}^*, \tilde{\mathbf{a}}_c^*)$ with $c \in \{1, \ldots, C\}$ to MILP$^{\mathrm{tr}}$-ϵM with objective function value $\tilde{z}^* < z^*$. Given a sufficiently large value of M, $(\tilde{a}_{c0}^* - \mathbf{u}\mathbf{U}^{-1}\tilde{\mathbf{a}}_c^*, \mathbf{U}^{-1}\tilde{\mathbf{a}}_c^*)$ is a feasible solution to MILP-ϵM with objective function value $\tilde{z}^* < z^*$. This contradicts that $(a_{c0}^*, \mathbf{a}_c^*)$ with $c \in \{1, \ldots, C\}$ is the optimal solution to MILP-ϵM. □

Appendix B

Take $(N, P, C) = (7, 1, 2)$. Patterns 1, 2, 3 and 4 belong to class 1 (represented by △), patterns 5, 6 and 7 to class 2 (represented by □). The measurements are $x_{11} = 1$, $x_{21} = 2$, $x_{31} = 3$, $x_{41} = 15$, $x_{51} = 4$, $x_{61} = 5$ and $x_{71} = 6$ and are shown in Fig. 8. Set $\epsilon = 1$. It is easy to check that for $M \leqslant 2(x_{41} - 3.5) + \epsilon = 24$ the optimal objective function value of MILP-ϵM is 1. Take M arbitrarily large but finite. Set $x_{41} > \frac{M-\epsilon}{2} + 3.5$. Given the value of M and the change made to x_{41}, all solutions with one misclassification are cut away now. Nevertheless, it is obvious that there still exist feasible solutions with only one misclassification provided the value of M would be sufficiently large. Pattern 4 is clearly a heavy outlier.

Appendix C

The LP relaxation of MILP-ϵM has the trivial solution $a_{cp} = a_{dp}$ for $c, d \in \{1, \ldots, C\}$ and $p \in \{0, 1, \ldots, P\}$ if and only if there exist values $\breve{u}_{nc} \in \mathrm{IR}^+$ such that

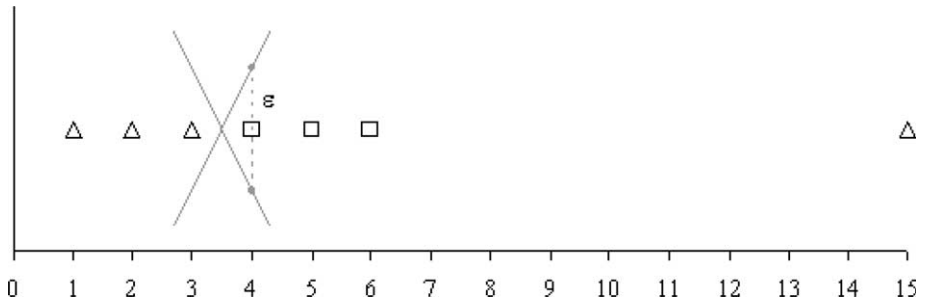

Fig. 8. Example of how to construct an instance for which any finite M is too small.

$$\sum_{c=1}^{C} \breve{u}_{nc} = 1, \quad n \in \{1,\ldots,N\},$$

$$N_c = \sum_{n=1\ c_n \neq c}^{N} \breve{u}_{nc}, \quad c \in \{1,\ldots,C\},$$

$$\sum_{n=1\ c_n=c}^{N} x_{np} = \sum_{n=1\ c_n \neq c}^{N} \breve{u}_{nc} x_{np}, \quad c \in \{1,\ldots,C\},\ \ p \in \{1,\ldots,P\}.$$

Proof. The trivial solution is optimal if and only if its objective function value equals the objective function value of a dually feasible solution. Hence, the trivial solution is optimal if and only if there exist values for $u_{nc} \in \mathrm{IR}^+$ such that

$$\frac{N\epsilon}{M} = \epsilon\left(\sum_{n=1}^{N}\sum_{c=1}^{C} u_{nc}\right), \tag{C.1}$$

$$M\left(\sum_{c=1}^{C} u_{nc}\right) \leqslant 1, \quad n \in \{1,\ldots,N\}, \tag{C.2}$$

$$\sum_{n=1\ c_n=c}^{N}\sum_{c=1}^{C} u_{nc} - \sum_{n=1\ c_n\neq c}^{N} u_{nc} = 0, \quad c \in \{1,\ldots,C\}, \tag{C.3}$$

$$\sum_{n=1\ c_n=c}^{N}\sum_{c=1}^{C} u_{nc}x_{np} - \sum_{n=1\ c_n\neq c}^{N} u_{nc}x_{np} = 0, \quad c \in \{1,\ldots,C\},\ \ p \in \{1,\ldots,P\}. \tag{C.4}$$

(C.1) is satisfied if and only if the N dual constraints in (C.2) are satisfied to equality. Using this and setting $\breve{u}_{nc} = u_{nc}M$, the optimality conditions for the trivial solution become

$$\sum_{c=1}^{C} \breve{u}_{nc} = 1, \quad n \in \{1,\ldots,N\}, \tag{C.5}$$

$$N_c = \sum_{n=1\ c_n\neq c}^{N} \breve{u}_{nc}, \quad c \in \{1,\ldots,C\}, \tag{C.6}$$

$$\sum_{n=1\ c_n=c}^{N} x_{np} = \sum_{n=1\ c_n\neq c}^{N} \breve{u}_{nc}x_{np}, \quad c \in \{1,\ldots,C\},\ \ p \in \{1,\ldots,P\}, \tag{C.7}$$

where $\breve{u}_{nc} \in \mathrm{IR}^+$. □

Appendix D

SPEC marks found at http://www.spec.org

Machine name	SPEC fp92	SPEC intfp92	SPEC fp95	SPEC intfp95
SPARCstation/server 20 Model 61 (60 MHz)	102.8	88.9	–	–
SPARCstation/server 5 (70 MHz)	47.3	57.0	–	–
SPARCstation 20 Model 151 (150 MHz)	–	–	4.71	4.02
SPARCstation 5 Model 170 (170 MHz)	–	–	3.00	3.53
Intel SE440BX2 Motherboard(550 MHz, Pentium III)	–	–	15.1	22.3

Ref. [17] reports CPU times in terms of DEC 3000 Alpha Model 300 (150 MHz) seconds. We suggest the following crude approximation to compare their CPU times with ours. All CPU times in [17] are multiplied by 1.4 to obtain the equivalent CPU times on a SPARCstation/server 20 Model 61 (60 MHz). The factor of 1.4 is used by [17], based on 92 SPEC marks considerations. The more modern version of the SPARCstation/server 20 Model 61 (60 MHz) is the SPARCstation 20 Model 151 (150 MHz). It differs in CPU speed and memory. Assume that a task that takes one second on the 60 MHz machine takes 60/150 seconds on a 150 MHz computer. The 95 SPEC marks for the SPARCstation 20 Model 151 (150 MHz) and a machine similar to ours, the Intel SE440BX2 Motherboard (550 MHz, Pentium III), are given in 4. A task that takes one second on the SPARCstation 20 Model 151 (150 MHz) takes about 4.4 seconds on our machine. This gives the following scaling factor: $1.4\times60/150\times4.4=2.5$. The same steps can be repeated with the SPARCstation/server 5 (70 MHz) and its more modern counterpart, the SPARCstation 5 Model 170 (170 MHz). This gives a scaling factor of $0.8\times70/170\times5.7=1.9$, which is of the same order of magnitude. The factor we will use is 2. The CPU times reported by [17] will be multiplied by two to allow for comparison with our CPU times.

References

[1] S.M. Bajgier, A.V. Hill, An experimental comparison of statistical and linear programming approaches to the discriminant problem, Decision Sciences 13 (1982) 604–618.
[2] W.J. Banks, P.L. Abad, An efficient optimal solution algorithm for the classification problem, Decision Sciences 22 (1991) 1008–1023.
[3] K.P. Bennett, O.L. Mangasarian, Multicategory discrimination via linear programming, Optimization Methods and Software 3 (1993) 27–39.
[4] K.P. Bennet, E.J. Bredensteiner, A parametric optimization method for machine learning, INFORMS Journal on Computing 19 (3) (1997) 311–318.
[5] C.L. Blake, C.J. Merz, UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine, CA, 1998.
[6] J.W. Chinneck, Fast heuristics for the maximum feasible subsystem problem, INFORMS Journal on Computing 13 (3) (2001) 210–223.
[7] R.O. Duda, P.E. Hart, D.H. Strok, Pattern Recognition, second ed., Wiley–Interscience, New York, 2001.
[8] S.S. Erenguc, G.J. Koehler, Survey of mathematical programming models and experimental results for linear discriminant analysis, Managerial and Decision Economics 11 (1990) 215–225.
[9] N. Freed, F. Glover, Simple but powerful goal programming models for discriminant problems, European Journal of Operational Research 7 (1981) 44–60.
[10] W.V. Gehrlein, General mathematical programming formulations for the statistical classification problem, Operations Research Letters 5 (6) (1986) 299–304.

[11] W. Gochet, A. Stam, V. Srinivasan, S. Chen, Multigroup discriminant analysis using linear programming, Operations Research 45 (2) (1997) 213–225.
[12] D.J. Hand, Construction and Assessment of Classification Rules, John Wiley, Chichester, 1997.
[13] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer, 2001.
[14] G.J. Koehler, Linear discriminant functions determined by genetic search, ORSA Journal on Computing 3 (4) (1991) 345–357.
[15] G.J. Koehler, S.S. Erenguc, Minimizing misclassifications in linear discriminant analysis, Decision Sciences 21 (1990) 63–85.
[16] J.M. Liittschwager, C. Wang, Integer programming solution of a classification problem, Management Science 24 (14) (1978) 1515–1525.
[17] T. Lim, W. Loh, Y. Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Machine Learning 40 (2000) 203–229.
[18] P. Marcotte, G. Marquis, G. Savard, A new implicit enumeration scheme for the discriminant analysis problem, Computers Operations Research 22 (6) (1995) 625–639.
[19] P.A. Rubin, Heuristic solution procedures for a mixed-integer programming discriminant model, Managerial and Decision Economics 11 (1990) 255–266.
[20] P.A. Rubin, Solving mixed integer classification problems by decomposition, Annals of Operations Research 74 (1997) 51–64.
[21] L. Schrage, LINDO: Optimization Software for Linear Programming, Lindo Systems, Chicago, IL, 1995.
[22] R.C. Soltysik, P.R. Yarnold, The Warmack–Gonzalez algorithm for linear two-category multivariable optimal discriminant analysis, Computers Operations Research 21 (7) (1994) 735–745.
[23] A. Stam, E.A. Joachimsthaler, A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem, European Journal of Operations Research 46 (1) (1990) 113–122.
[24] D. Tax, R. Duin, Using two-class classifiers for multiclass classification. Proceedings 16th International Conference on Pattern Recognition, Quebec City, Canada, vol. II, IEEE Computer Society Press, Los Alamitos, 2002, pp. 124–127.
[25] T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, J. Vandewalle, Benchmarking least squares support vector machine classifiers, Machine Learning, forthcoming.
[26] V.N. Vapnik, Statistical Learning Theory, John Wiley, New York, 1998.
[27] A. Webb, Statistical Pattern Recognition, Arnold London, London, 1999.