Solving mixed integer classification problems by decomposition

Paul A. Rubin
Department of Management, The Eli Broad Graduate School of Management, Michigan State University, East Lansing, MI 48824-1122, USA
E-mail: rubin@msu.edu

Research into the accuracy of mixed integer programming models for discrimination and classification, and the efficacy of heuristics developed for them, has been hampered by the inability to solve to optimality problems with moderate to large sample sizes. We present encouraging preliminary results for a decomposition approach that allows solution of models with dimensions previously considered prohibitive.

Keywords: discriminant analysis, mixed integer program.
Subject classification: AMS(MOS) 62H30, 65U05.

1. Introduction

Early methods of statistical classification and discrimination were based on assumptions of normality [7, 30]. Subsequently, a number of authors experimented with linear [5, 9, 10, 13, 14, 16, 18, 21, 26] and integer [1, 11, 12, 20, 32, 34] programming models for selecting discriminant functions; since then, more esoteric approaches, including nonlinear programming [3, 33] and neural networks [4, 25, 35, 36], have been applied. The bulk of this work (and the balance of this paper) focuses on finding linear discriminant functions for problems with two populations.

Although mixed integer classification models were among the earliest introduced [19, 29], the preponderance of the mathematical programming models proposed are linear programs. In some sense, this trend runs contrary to the nature of the problem. The purpose of discriminant analysis is to develop a function, or set of functions, which classifies points as accurately as possible based on observed attributes. When the functions are required to be linear, this relates closely to the geometric problem of separating the convex hulls of the different populations (or convex hulls containing significant portions of those populations) via one or more hyperplanes. All methods select functions which are "best" on training samples, hoping that this generalizes to the full populations. Mixed integer models seek to minimize the actual number (or cost) of errors in the training samples. Linear programs, in contrast, seek to

optimize some measure of the distance of each population's hull from the separating hyperplane(s), or the distance misclassified points lie on the wrong side of the hyperplane(s). Mixed integer models clearly attack the underlying goal of accurate classification more directly. That mixed integer approaches have not superseded linear programs seems likely to be due to the less tractable computational nature of the mixed integer models.

A number of seemingly efficacious heuristics have been proposed for mixed integer classification models [17,23,27]. In Monte Carlo studies, they have tended to fare reasonably well but have not excelled. It remains an open question to what extent this accrues from limitations in the models themselves versus any failure of the heuristics to develop fully optimal solutions to them. In order to test both the quality of the functions generated by those models and the accuracy of the heuristics in solving the models, it is necessary to solve to optimality test models of reasonable dimensions. As an alternative to standard branch-and-bound methods, several special-purpose algorithms have been proposed to optimize mixed integer classification models [1,17,31]. Published studies have largely been limited to combined sample sizes of approximately 100 observations when doing this, although Soltysik and Yarnold [31] solved one problem with 162 observations and Marcotte et al. [23] recently reported results up to sample size 300.

We present here a decomposition scheme for solving two-group mixed integer classification models that offers a first step toward alleviating this limitation. Preliminary tests suggest that it has the potential to allow solution of problems several times larger than was previously possible. In subsequent sections, we introduce the mathematical representation of the problem, develop the solution scheme, and evaluate the method on a few test problems. Since the validity and speed of the solution procedure are at issue here, rather than the accuracy of the classification functions produced, we do not employ hold-out samples in the computational evaluations.

2. A mixed integer classification model

The model we will use resembles several previously published. Assume that training samples of sizes N_1 and N_2 from two distinct populations are given, each sample point containing observations of K attributes. For the gth group (population), let $\mathbf{X}_g$ be the $N_g \times K$ matrix of observed attribute values, γ_g the cost of misclassifying an observation from that group, and π_g the prior probability of an observation coming from that group. We seek coefficients w_k ($k = 1,\ldots,K$) and c such that the function

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} w_k x_k - c$$

is negative when observation $\boldsymbol{x}$ comes from the first population and positive when it comes from the second population. Since perfect accuracy may not be achievable, we

solve the following mixed integer linear program to obtain a function which minimizes expected misclassification cost on the training samples:

$$\begin{aligned}
\text{minimize} \quad & \sum_{g=1}^{2} \frac{\gamma_g \pi_g}{N_g} (\mathbf{1}' \mathbf{z}_g) - \varepsilon d \\
\text{subject to} \quad & \mathbf{X}_1 \boldsymbol{w} - c\mathbf{1} + d\mathbf{1} - \mathbf{M}_1 \boldsymbol{z}_1 \leq \mathbf{0}, \\
& \mathbf{X}_2 \boldsymbol{w} - c\mathbf{1} - d\mathbf{1} + \mathbf{M}_2 \boldsymbol{z}_2 \geq \mathbf{0}, \\
& -1 \leq \boldsymbol{w} \leq 1, \\
& c \text{ free}, \\
& d \geq \delta, \\
& \boldsymbol{z}_1, \boldsymbol{z}_2 \text{ binary}.
\end{aligned} \tag{1}$$

Here, **1** is a vector of appropriate dimension, all of whose entries are unity. The binary variable z_{gn} indicates whether or not the nth observation from the gth group has been misclassified. Variable d measures the minimum extent by which scores of correctly classified observations differ from the critical value of zero. The term

$$\frac{1}{N_g} (\mathbf{1}' \mathbf{z}_g)$$

in the objective function estimates the probability of an observation from group g being misclassified, based on the frequency of misclassification within the training sample. The summation in the objective function is thus the total expected misclassification cost. As noted in [27], small positive constants δ and ε can be chosen such that the optimal value of d is distinguishable from zero (meaning that correctly classified points are classified unambiguously); this makes d useful as a "tie-breaker" among competing functions with equal misclassification rates, while not giving d sufficient weight in the objective to induce unnecessary misclassifications.[1)] Due to differences in the objective functions of [27] and this model, the coefficient ε derived in the former needs to be scaled here by

$$\max_{g=1,2} \left(\frac{\gamma_g \pi_g}{N_g} \right).$$

$\mathbf{M}_1$ and $\mathbf{M}_2$ are diagonal matrices whose diagonal coefficients are positive and sufficiently large to render the corresponding constraints nonbinding when the associated binary variables take value unity. Similar constructs are found in most if not all mixed integer classification models.

[1)] Regardless of the choice of δ, an observation $\boldsymbol{x}$ will be classified into the first group, the second group or neither according to the sign (−1, +1 or 0) of $f(\boldsymbol{x})$. The effect of δ is to treat observation $\boldsymbol{x}$ as misclassified in the objective function of (1) if its "score" $f(\boldsymbol{x})$ is too close to zero.

The most common method of solving mixed integer problems, the method which dominates "off-the-shelf" software solutions, is branch-and-bound, in which bounds are typically derived from the continuous (linear programming) relaxation of the model. The efficiency of branch-and-bound is tied directly to the tightness of those bounds. In the continuous relaxation of model (1), variable z_{gn} will be assigned the value

$$\frac{\left|\sum_{k=1}^{K} X_{gnk} w_k - c\right| + d}{M_{gn}},$$

assuming that the corresponding observation is not being classified correctly. Even with conservative choices of $\mathbf{M}_g$, these positive values for the relaxed variables are too close to zero to produce tight objective bounds. Thus, nodes in the search tree are difficult to fathom, the search progresses deeper into the tree, and for even modest sample sizes the process takes too long to complete.

3. Decomposition of the problem

Despite the poor quality of the bounds in the continuous relaxation of (1), advances in both computer hardware and branch-and-bound software make it possible to solve problems with combined sample sizes of 50 or more in a matter of seconds. This enables us to attack problem (1) by means of decomposition. Let us divide each sample into $P > 1$ disjoint subsamples, partitioning the $\mathbf{X}$ and $\mathbf{M}$ matrices correspondingly into $\mathbf{X}_{1p}$, $\mathbf{X}_{2p}$, $\mathbf{M}_{1p}$ and $\mathbf{M}_{2p}$ ($p = 1,\ldots,P$). We also clone the variables $\boldsymbol{w}$, c and d, partition the variables $\boldsymbol{z}$, and introduce the following modified version of (1):

$$\begin{array}{lllll}
\text{minimize} & \displaystyle\sum_{g=1}^{2}\sum_{p=1}^{P} \frac{\gamma_g \pi_g}{N_g}(\mathbf{1}'\boldsymbol{z}_{gp}) - \varepsilon d & & & \\
\text{subject to} & \mathbf{X}_{1p}\boldsymbol{w}_p - c_p\mathbf{1} + d_p\mathbf{1} - \mathbf{M}_{1p}\boldsymbol{z}_{1p} & \le \mathbf{0} & (p = 1,\ldots,P), & \\
& \mathbf{X}_{2p}\boldsymbol{w}_p - c_p\mathbf{1} - d_p\mathbf{1} - \mathbf{M}_{2p}\boldsymbol{z}_{2p} & \ge \mathbf{0} & (p = 1,\ldots,P), & \\
& -\mathbf{1} \le \boldsymbol{w}_p \le \mathbf{1} & & (p = 1,\ldots,P), & \\
& c_p \text{ free} & & (p = 1,\ldots,P), & (2)\\
& d_p \ge \delta & & (p = 1,\ldots,P), & \\
& \boldsymbol{z}_{1p}, \boldsymbol{z}_{2p} \text{ binary} & & (p = 1,\ldots,P), & \\
& \boldsymbol{w}_p - \boldsymbol{w}_{p-1} & = \mathbf{0} & (p = 2,\ldots,P), & \\
& c_p - c_{p-1} & = 0 & (p = 2,\ldots,P), & \\
& d_p - d_{p-1} & = 0 & (p = 2,\ldots,P). &
\end{array}$$

Problem (2) is mathematically equivalent to (1). The last three sets of equations impose consistency of the variables $\boldsymbol{w}$, c and d across all blocks. We can get a lower

bound for the objective value of (1) by relaxing the consistency constraints in (2), causing the problem to separate into P disjoint subproblems of the form of (1), with two changes: each subproblem involves only portions of the original samples; and the objective term εd is subtracted only in one subproblem, arbitrarily the first.

We now apply a depth-first branch-and-bound strategy to solving the overall problem. At each node, we solve the continuous relaxation of the full problem, and fathom the node if it is infeasible, if its relaxed objective value exceeds the current incumbent, or if it yields up an integer-feasible solution (recording the solution as a new incumbent if appropriate). If the node survives this step, we solve each of the subproblems *as a mixed integer program* (no relaxation of the integrality restrictions), adding their objective values. If this sum is larger than the objective result of the linear program, we make it the new lower bound for the node, and fathom the node if appropriate. Should the node survive again, we select a binary variable whose value in the continuous relaxation was not integer and separate the node based on that variable. In the computational trials discussed below, we chose the binary variable whose value was closest to 0.5, and after separation branched first to the child node in which that variable was fixed at unity.

One key to the efficacy of this approach is the selection of subproblem sizes. On the one hand, subproblems must be small enough so that they can be solved quickly, since a significant number of them will be solved in the course of the algorithm. On the other hand, the more subproblems we create, the weaker the sum of their objectives will be as a bound for the value of the full problem at that node. Note that if the continuous relaxation bound is tighter than the subproblem-generated bound too frequently, we have accomplished nothing other than to waste time; the algorithm devolves into simple depth-first branch-and-bound. For a given problem (and a given computing platform), a little experimentation may be needed to find a good value for P. In addition, it is desirable to keep all subproblems active in main memory concurrently. The software used, assuming it has this capability, may impose limits on the number of subproblems for which this can be done.

There are a few tactical opportunities to improve the speed of the algorithm. When a new node is created, we lock one of the binary variables z_{gn} at either zero or one in both the linear program and the subproblem to which it belongs. In addition, we take the corresponding constraint of (1), which can be written generically as

$$(-1)^{(g+1)}\left(\sum_{k=1}^{K} X_{gnk} w_k - c\right) + d - M_{gn} z_{gn} \leq 0,$$

and impose a version of it on every other subproblem. If z_{gn} is locked at zero, the added constraint is

$$(-1)^{(g+1)}\left(\sum_{k=1}^{K} X_{gnk} w_k - c\right) + d \leq 0; \tag{3}$$

if z_{gn} is locked at one, the new constraint is

$$(-1)^{(g+1)}\left(\sum_{k=1}^{K} X_{gnk} w_k - c\right) + d \geq 0, \tag{4}$$

reflecting the consideration that observation n of group g would not be misclassified unless its score had the wrong sign. The added constraints have the effect of tightening the contribution to the lower bound of subproblems other than the one containing the newly locked variable.

We can also greatly reduce the number of times we solve mixed integer problems by retaining the solutions from one node to the next. If the value of variable z_{gn} in the most recent solution to the subproblem which owns it matches the value at which we are locking it, there is no need to solve that problem again in the child node, as the same solution remains optimal. Similarly, if the most recent solution of any other subproblem satisfies whichever one of (3) and (4) we add to it, there is no need to solve that subproblem again in the child node: we may simply retain the previous solution.

We can make another, more modest, reduction in the number of mixed integer problems to be solved by noting two things about the bounding process. The first, and more obvious, is that the objective value of any subproblem (other than the first) must be nonnegative. (The first subproblem generates a negative objective value if and only if it misclassifies no points.) Thus, as we accumulate the partial sum of subproblem objective values, if at any point that sum equals or exceeds the value of the incumbent solution, we can immediately fathom the node, without solving any remaining subproblems. Second, when moving from a parent node to a child, each subproblem's objective value is nondecreasing. Thus, if any one subproblem's objective increases by at least the difference between the current incumbent and the sum of subproblem objectives at the parent node, we can immediately fathom the node: the overall sum of subproblem values will equal or exceed the incumbent.

Finally, we can improve progress by being a bit aggressive seeking new incumbents. We evaluate the discriminant functions arising from the values of $\boldsymbol{w}$ and c produced by the continuous relaxation and by each solved mixed integer subproblem, counting the number of points from the training samples they misclassify, and update the incumbent solution accordingly.

One of the charms of integer programming is that one model may solve rapidly while another version, only slightly modified, may solve quite slowly. Accordingly, it is prudent to impose either time or memory limits on the subproblems attempted. Moreover, since we anticipate solving quite a few subproblems, it is advisable to make those limits fairly parsimonious. Should our solver fail to optimize a subproblem within the allotted time and space, we simply retain the solution to that subproblem in the parent node, which gives a lower bound for its objective value at the child node. This helps control execution time, at the cost of loosening the bounds a little and possibly increasing the total number of nodes traversed.

3.1. Relationship to Lagrangian relaxation

The reader may wonder why the consistency constraints in (2) were simply dropped, rather than incorporating them as penalty terms in the objective function in the manner of Lagrangian relaxation [6]. Lagrangian relaxation produces the same number of subproblems, of the same size and difficulty. Indeed, dropping the consistency constraints is equivalent to relaxing them into the objective using multiplier values all equaling zero.

We did in fact set out initially to use Lagrangian relaxation. In our early tests, though, zero multipliers always produced the tightest bounds, with the bounds falling off as we tried to improve the multipliers. This held true even when we used the more computationally intensive line search method for optimizing the dual function, rather than the more common discrete step approach. While we do not have a certain explanation for this phenomenon, we believe it is likely due to the frequent existence of multiple subproblem solutions of equal classification accuracy. When multiple solutions exist, any attempt to adjust the Lagrange multipliers in the direction of the consistency violations may cause one of the alternative optima to rise to the fore, and the subproblem objective to drop rather than to increase. We have no particular reason to expect the zero multiplier vector to be optimal in the Lagrangian dual in general, but if it is not, either the direction suggested by the consistency violations is not an improving direction for the dual function, or numerical instability forces extremely small step sizes. Whatever the cause, the effect was that we never saw a bound tighten after a Lagrangian step, and hence dropped the attempt at multiplier improvement.

4. Computational trials

We have tested the decomposition procedure on several data sets, with encouraging results. All trials were done on a personal computer containing a 90 MHz Pentium processor and 16 megabytes of random access memory. Model files were generated using the AMPL modeling language [8] and solutions were attempted using the CPLEX mixed integer solver [37]. Our algorithm was coded as a FORTRAN program that calls the CPLEX mixed integer library. The algorithm is detailed in the appendix. We also used a version of the MultiODA method [31], again coded in FORTRAN. MultiODA's subproblems are systems of linear equations, rather than linear programs. To solve those systems, we employed source code drawn from Kahaner et al. [15], retrieved from the NIST Guide to Available Mathematical Software (GAMS) repository [2]. The Warmack–Gonzalez algorithm [38], which underlies MultiODA, is not well suited to dealing with nominal or ordinal data, which our first two test problems contained. We modified it slightly to overcome that limitation [28].

Data sets were drawn from files obtained at the UCI Machine Learning repository [24]. Their characteristics are summarized in table 1. We were concerned with neither the accuracy of the model nor the speed with which different methods first

Table 1

Data sets.

Property	Flags (F)	Breast cancer I (BC I)	Breast cancer II (BC II)
Group 1 size (N_1)	39	240	444
Group 2 size (N_2)	155	160	239
Attributes (K)	7	18	9
Binary attributes	5	18	0

Table 2

Results.

Property	Data set	Decomposition	Branch-and-bound	MultiODA
Execution time	F	00:34	10:00	14:00
(hours:minutes)	BC I	02:20	08:58	11:19
	BC II	01:24	06:10	11:35
Objective value	F	**0.1701**	**0.1701**	0.1804
(**bold** = optimal)	BC I	**0.0600**	**0.0600**	0.1750
	BC II	**0.0161**	**0.0161**	0.1127
Completed?	F	Yes	No (time)	No (time)
(proving optimality)	BC I	Yes	No (time)	No (time)
	BC II	Yes	No (memory)	No (time)

located an optimal discriminant function (although the latter proved to be comparable between our method and the CPLEX solver). Our focus was instead on the speed with which each method proved optimality. We used equal misclassification costs ($\gamma_1 = \gamma_2 = 1$), and set the prior probabilities (π_1, π_2) equal to the proportions of the combined training sample belonging to each group.

Results of the computational tests are listed in table 2. Applications of our decomposition algorithm were preceded by short trial runs (approximately five minutes aggregate execution time per data set) to evaluate subproblem solution speeds. By watching progress of the lower bound over a few iterations at each of two or three subproblem sizes (dictated by the number of subproblems P), the user will quickly get a sense of what a good (though not necessarily optimal) choice for P might be.

4.1. World's flags

Our first tests used data on the flags of all the world's nations (data set donated by R.S. Forsyth). Records of 194 national flags were divided into two groups, 39 (N_1)

whose emblems contained animate objects (animals, plants or human body parts) and 155 (N_2) whose emblems did not. Three attributes of the countries were chosen: continent (counting Oceania as a sixth "continent"); land area; and population. The latter two variables were scaled to minimize numerical precision problems reported by the software; this scaling truncated a number of the smaller areas and populations to zero. The nominal continent variable was replaced with five indicator variables, for a total of seven (K) attribute variables (five of them binary).

Attempts to solve the problem using standard branch-and-bound failed. We initially employed CPLEX using a breadth-first search strategy, but found after approximately half an hour no discernible advance of the global lower bound on the objective. In addition, nodes were accumulating at a troublesome rate; when the tree grows too large, the computer bogs down swapping pieces of it into and out of secondary storage. We restarted the solver using a depth-first strategy, which controls the number of nodes, with an initial incumbent of 0.175 (found in an earlier run). The solver found an integer feasible solution with value 0.1701 early in the process, but after 10 hours and 3.7 million nodes, it still had not completed the search. Unfortunately, with depth-first search there is no way of knowing how close the solver is to being finished, but we did observe it spending most of its time in portions of the tree 25 to 35 levels deep.

We applied the MultiODA procedure to this data set, but terminated the run after 14 hours, at which time it had performed approximately 32.45 million iterations. The best solution it had found to that point had an objective value of 0.1804 (found after approximately 5.45 million iterations). As with CPLEX, there is no indication how close MultiODA was to completing the search.

We then applied our algorithm to the problem, splitting it into four (P) subproblems of approximately 50 binary variables each, with each group represented proportionally in each subproblem. Our algorithm completed the search in 34 minutes, investigating 1,368 nodes; it confirmed that 0.1701 was indeed the optimal solution.

4.2. Breast cancer data I

A second test was performed using data collected by Dr. W.H. Wolberg of the University of Wisconsin Hospitals, Madison, and provided to the repository by Mangasarian [22]. The data represent measurements, on a 10-point scale, of nine different attributes of cellular structures; the groups consist of benign and malignant growths. We generated a random subsample of 400 observations from the full data set, of which 240 (N_1) came from the benign group and 160 (N_2) from the malignant group, in approximately the same proportions as the total data set. Treating the attribute coding as nominal, we selected two of the nine attributes (clump thickness and marginal adhesion) and recoded each using nine indicator variables, for a total of 18 (K) binary attribute variables.

We began by attempting to optimize the full model using branch-and-bound with best-bound (breadth-first) node selection. CPLEX generated an incumbent value of

0.0598 (which later proved to be optimal) in roughly 13 minutes; after nine hours, that remained the incumbent, the program having tested 956 thousand nodes, with 46 thousand nodes still active when we terminated the search. The best node bound at that point was 0.0076. We then employed MultiODA, which we terminated after 11 hours (14.4 million iterations). The incumbent solution, with an objective value of 0.1750, had been found at iteration 28. Last, we applied our algorithm, using two (P) subproblems. The small number of subproblems was prompted by two short pilot runs, which demonstrated that subproblems containing 100 observations each solved in acceptable times using this data, while bounds produced using four or five subproblems frequently were looser than those from the continuous relaxation. Our program completed the search in two hours and 20 minutes, with an optimal objective value of 0.06. The slight discrepancy between this and the breadth-first solution, due to the accuracy criterion we used in fathoming nodes, is less than the cost of one misclassification. Our program investigated a total of 158 nodes.

4.3. Breast cancer data II

In our final test, we next tried to solve a problem containing all 683 useable observations from the breast cancer data set (eliminating the remaining observations due to missing values). This sample contained 444 (N_1) observations from the benign group and 239 (N_2) from the malignant group. We used all nine (K) attributes, this time treating them as interval data. Following a short trial run, we decided to split the problem into five (P) subproblems. Our program completed the search, with an optimal objective value of 0.0161, in one hour and 24 minutes, testing 74 nodes. CPLEX, again using best-bound search, ran for slightly more than six hours before the search tree exhausted our imposed memory limit of 20 megabytes. At that point, CPLEX had visited more than 43 thousand nodes, of which almost 30 thousand were still alive. It had an incumbent solution of 0.0161 but a best node (lower) bound of only 0.0088. We allowed MultiODA to run for 11.5 hours (6.1 million iterations). Its incumbent solution when we terminated it, with objective value 0.1127, had been found after approximately 4.2 million iterations.

5. Conclusions

The trials described above demonstrate that the decomposition method provides a viable alternative to both standard branch-and-bound procedures and existing special-purpose algorithms for solving mixed integer classification models. The decomposition method supplies both a solution and a guarantee of optimality. While it will always be possible to generate problems too large for any combination of algorithm and computing platform to solve, the decomposition technique extends significantly the size of problems we are capable of solving today.

As with any branch-and-bound method, the decomposition algorithm may be aided by supplying a good incumbent solution, perhaps generated by a heuristic

method, at the outset. In addition, we suspect (but have not yet investigated) that the bounds generated by the decomposition algorithm could be tightened if observations were assigned to subproblems in proportion not just to their groups but to the segments of their groups classified correctly/incorrectly by the optimal discriminant function (or, as its surrogate, a good function generated by a heuristic method). For instance, if 30% of the population came from the first group, and if a heuristic solution classified 80% of the first group and 70% of the second group correctly, then we would attempt to assign observations to subproblems so that each subproblem drew approximately 24% of its points from members of the first group classified correctly by the heuristic, 6% from incorrectly classified members of the first group, 49% from correctly classified members of the second group, and 21% from incorrectly classified members of the second group. By spreading the "difficult" points evenly through the subproblems, we would hope to raise the minimum bound contribution from each subproblem.

Appendix

In describing the decomposition algorithm, we assume that the reader is familiar with branch-and-bound methods and the associated terminology.

Step 1. *Initialization*

Step 1.1. Read the master problem (1) and set up its linear programming relaxation, denoted (LR).

Step 1.2. Select the number P of subproblems and partition into P mixed integer subproblems, denoted (S_1) through (S_P).

Step 1.3. Set the upper bound $f_u := \infty$, and the initial lower bound $f_l(-1) := 0$. Set $depth := 0$. Flag all subproblems as unsolved.

Step 2. *Linear relaxation*

Step 2.1. Solve (LR), obtaining objective value f and values $\boldsymbol{z}$ for the (relaxed) integer variables.

Step 2.1.1. If (LR) is infeasible, go to step 4.

Step 2.2. Set $f_l(depth) := \max(f, f_l(depth - 1))$.

Step 2.3. Test the accuracy of the discriminant function produced by (LR) to see if it yields a new incumbent (reducing f_u).

Step 2.4. If $f_l(depth) \geq f_u$ or $\boldsymbol{z}$ is integer feasible, go to step 4.

Step 3. *Decomposition bound*

Step 3.1. Set $bound(depth) := 0$.

Step 3.2. For $p := 1 \ldots P$:

Step 3.2.1. If (S_p) is flagged unsolved, solve it; otherwise, use the retained previous solution. Denote the objective value $f_p(depth)$.

Step 3.2.1.1. If solution fails due to infeasibility, go to step 4.

Step 3.2.1.2. If solution fails due to time or memory limits, use the retained solution from the previous depth. (If failure occurs at depth 0, quit and go home.)

Step 3.2.1.3. If solution succeeds, flag (S_p) as solved. Test the accuracy of the discriminant function produced by (S_p) to see if it yields a new incumbent (reducing f_u).

Step 3.2.2. Set $bound(depth) := bound(depth) + f_p(depth)$.

Step 3.2.2.1. If $bound(depth) \geq f_u$, go to step 4.

Step 3.2.2.2. If $bound(depth - 1) + f_p(depth) - f_p(\text{depth} - 1) \geq f_u$, go to step 4.

Step 4. *Separation*

Step 4.1. Find the variable z_0 in the solution of (LR) having the worst integer infeasibility. Locate the subproblem (S_k) containing z_0. Let $\boldsymbol{x}_0$ denote the observation corresponding to z_0.

Step 4.2. Separate the current node, creating two new nodes at depth $depth + 1$.

Step 4.2.1. In the first child node:

Step 4.2.1.1. Fix z_0 at 1 in (LR).

Step 4.2.1.2. Fix z_0 at 1 in (S_k).

Step 4.2.1.3. For $p \neq k$, add to (S_p) the constraint $\boldsymbol{x}_0'\boldsymbol{w}_p - c_p + d_p \sim 0$, replacing $\sim$ with $\geq$ if $\boldsymbol{x}_0$ is in the first group and $\leq$ otherwise.

Step 4.2.1.4. For $p = 1,\ldots,P$ (including $p = k$), flag subproblem (S_p) as unsolved if the last recorded solution of it violates the new constraint.

Step 4.2.2. Proceed as above for the second child node, but fix z_0 at 0 and reverse the inequalities.

Step 4.3. Delete the current node, increment depth, select the first child node, and return to step 1.

Step 5. *Fathoming*

Step 5.1. Delete the current node.

Step 5.2. If another node exists at the current depth, select it and return to step 1.

Step 5.3. Otherwise, decrement *depth* repeatedly until either $depth = 0$ or a live node is found.

Step 5.3.1. If a live node is found, return to step 1.

Step 5.3.2. If $depth = 0$, stop; the algorithm has completed.

References

[1] W. Banks and P. Abad, An efficient optimal solution algorithm for the classification problem, Dec. Sci. 22(1991)1008–1023.

[2] R.F. Boisvert, S.E. Howe, D.K. Kahaner and J.L. Springmann, The guide to available mathematical software, NISTIR 90–4237, National Institute of Standards and Technology, Washington, DC, 1990.

[3] T. Cavalier, J. Ignizio and A. Soyster, Discriminant analysis via mathematical programming: On certain problems and their causes, Comp. Oper. Res. 16(1989)353–362.

[4] S.P. Curram and J. Mingers, Neural networks, decision tree induction and discriminant analysis: An empirical comparison, J. Oper. Res. Soc. 45(1994)440–450.

[5] S. Erenguc and G. Koehler, Survey of mathematical programming models and experimental results for linear discriminant analysis, Manag. Dec. Econ. 11(1990)215–225.

[6] M.L. Fisher, The Lagrangian relaxation method for solving integer programming problems, Mgt. Sci. 27(1981)1–18.

[7] R. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugenics 7(1936) 179–188.

[8] R. Fourer, D.M. Gay and B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Scientific Press, 1993.

[9] N. Freed and F. Glover, A linear programming approach to the discriminant problem, Dec. Sci. 12(1981)68–74.

[10] N. Freed and F. Glover, Simple but powerful goal programming models for discriminant problems, Eur. J. Oper. Res. 7(1981)44–60.

[11] W. Gehrlein, General mathematical programming formulations for the statistical classification problem, Oper. Res. Lett. 5(1986)299–304.

[12] W. Gehrlein and V. Gempesaw, A cost minimization approach to classification, in: *Proceedings of the 1991 Annual Meeting of the Decision Sciences Institute*, S. Melnyk, ed., Decision Sciences Institute, Atlanta, 1991, pp. 1162–1163.

[13] F. Glover, S. Keene and R. Duea, A new class of models for the discriminant problem, Dec. Sci. 19(1988)269–280.

[14] D. Hand, *Discrimination and Classification*, Wiley, 1981.

[15] D. Kahaner, C. Moler and S. Nash, *Numerical Methods and Software*, Prentice-Hall, 1989.

[16] M. Kendall, Discrimination and classification, in: *Multivariate Analysis: Proceedings of the International Conference on Multivariate Analysis*, P. Krishnaiah, ed., Academic Press, 1966, pp. 165–185.

[17] G. Koehler and S. Erenguc, Minimizing misclassifications in linear discriminant analysis, Dec. Sci. 21(1990)63–85.

[18] K.F. Lam, E.U. Choo and J.W. Moy, Minimizing deviations from the group mean: A new linear programming approach for the two-group classification problem, Eur. J. Oper. Res. 88(1996) 358–367.

[19] J. Liittschwager and C. Wang, Integer programming solution of a classification problem, Mgt. Sci. 24(1978)1515–1525.

[20] C. Loucopoulos and R. Pavur, Mixed integer programming approaches to the statistical classification problem for more than two groups, in: *Proceedings of the 1990 Annual Meeting of the Decision Sciences Institute*, B. Whitten and J. Gilbert, eds., Decision Sciences Institute, Atlanta, 1990, pp. 179–181.

[21] O.L. Mangasarian, Linear and nonlinear separation of patterns by linear programming, Oper. Res. 13(1965)444–452.

[22] O.L. Mangasarian and W.H. Wolberg, Cancer diagnosis via linear programming, SIAM News 23, 5 (1990) pp. 1, 18.

[23] P. Marcotte, G. Marquis and G. Savard, A new implicit enumeration scheme for the discriminant analysis problem, Comp. Oper. Res. 22(1995)625–639.
[24] P.M. Murphy and D.W. Aha, 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. 1994.
[25] E. Patuwo, M.H. Hu and M.S. Hung, Two-group classification using neural networks, Dec. Sci. 24 (1993)825–845.
[26] P. Rubin, Evaluating the maximize minimum distance formulation of the linear discriminant problem, Eur. J. Oper. Res. 41(1989)240–248.
[27] P. Rubin, Heuristic solution procedures for a mixed-integer programming discriminant model, Manag. Dec. Econ. 11(1990)255–266.
[28] P. Rubin, Adapting MultiODA and the Warmack–Gonzalez algorithm to handle discrete data, Working paper, Department of Management, Michigan State University, East Lansing, MI, 1997.
[29] E. Saniga and W. Gehrlein, Some mathematical programming approaches to discrimination, in: *Proceedings of the Business and Economics Section of the American Statistical Association*, American Statistical Association, Washington, DC, 1978, pp. 623–624.
[30] C. Smith, Some examples of discrimination, Ann. Eugenics 13(1947)272–282.
[31] R.C. Soltysik and P.R. Yarnold, The Warmack–Gonzalez algorithm for linear two-group multivariable optimal discriminant analysis, Comp. Oper. Res. 21(1994)735–745.
[32] A. Stam, Extensions of mathematical programming-based classification rules: A multicriteria approach, Eur. J. Oper. Res. 48(1990)351–361.
[33] A. Stam and E. Joachimsthaler, Solving the classification problem in discriminant analysis via linear and nonlinear programming methods, Dec. Sci. 20(1989)285–293.
[34] A. Stam and E. Joachimsthaler, A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem, Eur. J. Oper. Res. 46 (1990)113–122.
[35] V. Subramanian, M.S. Hung and M.Y. Hu, An experimental evaluation of neural networks classification, Comp. Oper. Res. 20(1993)769–782.
[36] K.Y. Tam and M.Y. Kiang, Managerial applications of neural networks: The case of bank failure predictions, Mgt. Sci. 38(1992)926–947.
[37] *Using the CPLEX Callable Library*, CPLEX Optimization, Inc., Incline Village, NV, 1995.
[38] R.E. Warmack and R.C. Gonzalez, An algorithm for the optimal solution of linear inequalities and its application to pattern recognition, IEEE Trans. Comp. C-22(1973)1065–1075.