Misclassification Minimization: Exact and Heuristic Approaches

Pierre Hansen Alejandro Karam

This draft: August 17, 2005

Abstract

We consider the problem of separating two sets of points in an Euclidean space, with a hyperplane that minimizes the number of points lying on the "wrong" side of the plane. This problem is NP-complete, and only relatively small instances can be tackled with exact algorithms. We present and test an improved exact formulation. We also propose and test a heuristic approach, based on the Variable Neighborhood Search framework, to determine the plane coefficients. Numerical experiments with instances of up to 100,000 points in 6 dimensions, suggest that the heuristic method is fast and reliable. The discriminating hyperplanes found appear to generalize well on holdout sets, as compared to those obtained by alternative procedures.

1 Introduction

Given two distinct sets of points in an Euclidean space, we consider the problem of finding a hyperplane such that each half space defined by it is assigned to one of the sets, and the number of points lying in the half-space corresponding to the other set is minimized.

This problem is known to be NP-complete [MS92], and several approaches have been proposed to tackle it, both exactly and heuristically (e.g. [LW78, Geh86, KE90, MS92, MMS95, Rub97, SS97]).

2 Formulation

2.1 Notation and Statement of the Problem

The scalar product of two vectors x and y both in $\mathbb{R}^n$, is denoted $x^t y$. If M is a matrix, M_i represents its i^{th} row. When necessary, 0 and 1 denote, respectively, vectors of zeros and ones in appropriate dimensions. We denote $\mathcal{A}$ and $\mathcal{B}$ the two sets of points in $\mathbb{R}^n$ to be (approximately) separated by

the desired hyperplane. Their coordinates are represented by the matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{k \times n}$. For a given plane

$$P = \left\{ x \,\middle|\, w^t x = \gamma \right\} \text{ with } \gamma \in \mathbb{R}, w \in \mathbb{R}^n, w \neq 0$$

we assign the half planes $w^t x < \gamma$ to the set $\mathcal{A}$ and $w^t x > \gamma$ to $\mathcal{B}$. A point $x \in \mathcal{A} \cup \mathcal{B}$ is then said to be misclassified when

$$\begin{array}{ll} w^t x > \gamma & \text{if } x \in \mathcal{A}, \\ & \text{or} \\ w^t x < \gamma & \text{if } x \in \mathcal{B}. \end{array}$$

and the objective is to find w and γ such that the number of such points from both sets is minimized.

Most models used in the literature for misclassification minimization are variations of the following mixed integer program:

$$\min_{w,\gamma,y,z} \left\{ \sum_{i=1}^{m} y_i + \sum_{j=1}^{k} z_j \,\middle|\, \begin{array}{c} -w^t A_i + \gamma \leq M y_i \text{ for } i = 1, .., m \\ w^t B_j - \gamma \leq M z_j \text{ for } j = 1, .., k \\ y \in \{0,1\}^m, z \in \{0,1\}^k, w \in \mathbb{R}^n, \gamma \in \mathbb{R} \end{array} \right\} \quad (1)$$

where M is the famous "big M", a sufficiently large constant. The binary vectors y and z track misclassified points from sets $\mathcal{A}$ and $\mathcal{B}$, respectively. We will refer to this basic model in our subsequent discussion.

For practical classification applications, in particular when off-sample generalization is the main concern, it might be important to consider other criteria in the objective function. Key examples include asymmetrical weighting of misclassifications from each set (to take account of different error costs or priors, e.g. [LW78]) or introducing secondary criteria (such as considering deviations from the plane, e.g. [BH82, Rub97, PWL97]). We will not be concerned with these variations and, for comparability, will only consider the basic objective of minimizing the total number of misclassified points.

2.2 Background

In addition to the general challenges of any large, mixed integer programming task, exact approaches to the linear misclassification minimization must deal with specific problems which are also present in some continuous, LP-based classification models [Koe90]. Of these problems, perhaps the most pervasive is the issue of the null solution: in many formulations, an optimal solution can be $[\gamma w] = 0$, which is useless for classification purposes. A common practice to rule out this undesirable outcome is to impose an additional standardization constraint on the model. The choice of this constraint is, however, delicate, and a large part of the literature on LP-based approaches to classification has been devoted to this problem [Gri72, MM85, FG86, GKD88, CIS89, Koe89a, Koe89b, Glo90, Koe91, Xia93, Gle99].

Many contributions in the literature concentrate on the algorithmic challenges of the MIP, without due attention to perverse, unforeseen consequences of the choice of standardization constraint. For example, [MS92] impose[1] $w_1 = \pm 1$, while [MMS95] postulate $\max(|w_1|, |w_2|, \ldots, |w_n|, \gamma) = 1$. Any of these constraints may preclude the optimal solution[2].

Another approach to deal with the null solution is to minimize the aggregate deviations from two reference planes, one for each class, instead of one[3] [Smi68, Han81, BM92]. Although it has been applied for the misclassification minimization problem [Geh86, Rub90, PWL97], this approach is not in fact equivalent to minimizing the number of points and might yield a suboptimal solution[4].

Other approaches to deal with the null solution problem found in the literature include the use of secondary objectives [Rub97] (which can also stir the solution away from the desired optimum) or the imposition of arbitrary constraints on the objective function [KE90].

Some of these contributions, despite eventual formulation failures with respect to the null solution issue, introduce algorithmic improvements for the solution of the mixed integer programs, such as decompositions [Rub97] and branching strategies [SS97]. These techniques could in principle be applied to various valid formulations, but we shall not consider them here; the focus of our discussion and proposition for exact approaches will be on formulation. Our only digression on MIP solution issues will be the use of our heuristic bounds to accelerate a (generic) branch-and-bound process (see section XXX below).

Mangasarian [Man99] exposed a subtle relationship between the standardization constraint and the L_p-norm in which distances are considered. In particular, the L_p-norm distance between between a point $x \in \mathcal{A} \cup \mathcal{B}$ and its projection $\pi(x)$ to the plane P is given by

$$\|x - \pi(x)\|_p = \frac{|w^t x - \gamma|}{\|w\|'_p}$$

where $\|\cdot\|'_p$ denotes the dual norm[5] of $\|\cdot\|_p$. A convenient and perfectly valid

[1] This idea is akin to that proposed for LP-models in [FG86], where $\gamma = \pm 1$

[2] The constraint in [MS92] forbids (potentially optimal) solutions with $\gamma = 0$. The one in [MMS95] forces the solution $[\gamma \quad w]$ to lie in the L_1-norm unit sphere in $\mathbb{R}^{n+1}$; a potentially optimal solution with $\gamma > 1$ and $\max(|w_1|, |w_2|, ..., |w_n|) < 1$, not lying on this sphere, is thus excluded.

[3] For a survey and discussion of these double plane models, see CITE MON CHAPITRE SUR CREDIT SCORING.

[4] To see why this is so, suppose the optimal solution to the double-plane model has been found, and consider the region *between* the reference planes. Any point in this region will not be counted as misclassified in the objective function, but can very well be on the wrong side of the actual discriminating plane P, which lies midway between the two.

[5] We recall that, for $1 < p < \infty$, the dual norm $\|\cdot\|'_p = \|\cdot\|_q$ where $\frac{1}{p} + \frac{1}{q} = 1$. The cases $p = 1$ and $p = \infty$, are defined by a limit argument as $\|\cdot\|'_1 = \|\cdot\|_\infty$ and $\|\cdot\|'_\infty = \|\cdot\|_1$.

standardization choice suggested by this equation is then $\|w\|'_p = 1$ for an appropriate choice of p. This constraint does not rule out any direction for the plane, because the gradient w of the plane can lie anywhere on the surface of the L_p-norm unit sphere, and γ is unconstrained, thus permitting any offset with respect to the origin[6] The choice of the norm p might be of interest in some classification applications[7], but for the misclassification minimization problem, the only relevant cases are $p = 1$ and $p = \infty$, since they are manageable by linear formulations.

One of the earliest formulations for the misclassification minimization problem is due to Liittschwager and Wang [LW78]. Their standardization constraint is $\|w\|'_1 = \|w\|_\infty = 1$. This formulation (which we shall refer to as LW) has stood the test of time; it has been used as base for other research efforts (e.g. [SS97]) and, by the above discussion, unlike many of its successors, it is formally correct in that it does not exclude valid solutions from consideration as it avoids the null solution. Our formulation is an improvement on this model[8].

The LW formulation enforces the condition $\|w\|_\infty = 1$ within a single MIP, with the constraints

$$-1 + 2D_l \leq w_l \leq 1 - 2E_l \qquad l = 1, \ldots, n \tag{2}$$

$$\sum_{l=1}^{n} D_l + \sum_{l=1}^{n} E_l \tag{3}$$

$$E \in \{0,1\}^n \quad D \in \{0,1\}^n$$

The condition $\|w\|_\infty = 1$ can also be tackled by solving $2n$ separate programs, each of which forces w_l to be either 1 or -1, for $l = 1, \ldots, n$, and with $-1 \leq w_l \leq 1$. This is the preferred approach when minimizing the sum of L_1-norm distances [Man99]. We performed some experiments to compare the solution speed on the same set of problems, with the two approaches: a single MIP or $2n$ smaller MIPs. We found that there appears to be no advantage in general to splitting the problem into $2n$ slightly smaller ones, and we concentrate our subsequent discussion on the formulation with a single MIP. The issue, however, might deserve further study.

In [LW78], it is shown that the"big M" can be set to

$$M = 2n \max \left(\max_{\substack{i=1,\ldots,m \\ l=1,\ldots,n}} |A_{il}| \quad , \quad \max_{\substack{j=1,\ldots,k \\ l=1,\ldots,n}} |B_{jl}| \right) \tag{4}$$

[6]This idea had been applied for the Euclidean distance [CIS89], and elegantly generalizes to the arbitrary-norm case in [Man99].

[7]See, CITE MON PREMIER CHAPITRE, where the potential effect of the choice of p on off-sample generalization is explored.

[8]In [LW78] the y_i and z_i in the objective function are weighted by cost and prior probabilities. As explained in section 2.1, we ignore these weights.

We will assume that the data in A and B have been rescaled to the range $[0, 1]$; this is a quite standard procedure in practice, and all of our databases conform to it.

2.3 Our Model

A remarkable insight in [MMS95] is that the "big M" can in fact be determined on a *per observation* basis, thus resulting in a tighter formulation. We apply this idea to the LW framework, and use the constants

$$\begin{aligned} M_i^A &= n + n \max_{l=1,\ldots,n} |A_{il}| \qquad i = 1, \ldots, m \\ M_j^B &= n + n \max_{l=1,\ldots,n} |B_{jl}| \qquad j = 1, \ldots, k \end{aligned} \tag{5}$$

in the corresponding constraints, instead of the common M. The validity of this bound can be seen by an argument almost identical to that presented in [LW78], recalling that the matrices A and B have been rescaled[9].

Our final formulation is then

$$\min_{w,\gamma,y,z} \left\{ \sum_{i=1}^{m} y_i + \sum_{j=1}^{k} z_j \;\middle|\; \begin{array}{c} -w^t A_i + \gamma \leq M_i^A y_i \text{ for } i = 1, .., m \\ w^t B_j - \gamma \leq M_i^B z_j \text{ for } j = 1, .., k \\ -1 + 2D_l \leq w_l \leq 1 - 2E_l \\ \sum_{l=1}^{n} D_l + \sum_{l=1}^{n} E_l \end{array} \right\} \tag{6}$$

with $y \in \{0,1\}^m, z \in \{0,1\}^k, w \in \mathbb{R}^n, \gamma \in \mathbb{R}$, $E \in \{0,1\}^n$, $D \in \{0,1\}^n$ and the constants M_i^A and M_i^B defined as in (5).

3 Heuristic Approach

The basic idea of our approach is to decompose the problem into determining a direction for the hyperplane and finding the optimal position, of offset to the origin, in a given direction. This kind of decomposition is not uncommon in the global optimization literature [EXAMPLE], and has been applied to the misclassification minimization problem elsewhere [CM96]. The two main components of our method are thus:

- A function (henceforth referred to as "the oracle") which, given a direction for the plane, finds an optimal parallel shift and returns the corresponding number of misclassified points, and
- An optimization framework that searches the space of directions for a minimum, by repeatedly querying the oracle.

[9] The first n covers the worst case for γ and, e.g. for the case of $\mathcal{A}$, the term $n \max_{l=1,\ldots,n} |A_{il}|$ covers the highest possible absolute value of $w^t A_i$.

The space of directions is searched with a heuristic framework known as Variable Neighborhood Search (VNS) [HM98], which has been applied successfully in a number of global optimization tasks (e.g. [BM96, CH97, HM97, BCK99, HMU00, MPKVC00] [ARE THESE THE BEST EXAMPLES?]).

We first describe the oracle and then summarize the VNS framework within which it is inserted.

3.1 The Oracle

A plane P can be characterized either by the n components of its gradient w and the scalar γ or by $n-1$ angles $\alpha = (\alpha_1, \alpha_2, ..., \alpha_{n-1})$ and an offset constant φ. We choose to search the space of directions as characterized by the $n-1$ angles α. We thus avoid scaling issues in the choice of the gradient[10] and exploit in the algorithm the fact that φ represents directly the offset with respect to the origin[11]. The conversion from $[\alpha \quad \varphi]$ to the corresponding $[w \quad \gamma]$ can be performed by an iterative procedure [Ken61] [Som58].

The input to the oracle is a direction α from the origin. We recall the convention that the set $\mathcal{A}$ is meant to lie on the side of the plane to which w points. The output COUNTMISSED(α) is the number of points misclassified by the best possible plane along the given direction α. The algorithm is outlined in Figure 1.

The candidate plane, perpendicular to the ray, is initially positioned at the first point along the ray, i.e. the one with minimum distance. All the observations start thus on the same side of this plane, with $\mathcal{A}$ being in the correct half-space, while all $\mathcal{B}$ points are misclassified. The corresponding counters $missedA = 0$ and $missedB = |\mathcal{B}|$ are initialized. The initial number of misclassified points *totalmissed* corresponds to all the points in $\mathcal{B}$.

The plane is then moved along the ray, in the direction determined by α, to the position corresponding to each successive point. The counters $missedA$ and $missedB$ are updated for the current position depending on whether it corresponds to a point in $\mathcal{A}$ or $\mathcal{B}$. The minimum number *best* found along the way is returned. Its position is also recorded.

The oracle function COUNTMISSED(α) can then be queried by the optimization routine as explained below. Note that the angles (in radians) given by α are automatically reduced modulo 2π within the oracle[12]. This implies that the optimization routine can search over the unconstrained domain $\alpha \in \mathbb{R}^{n-1}$.

[10] We recall that there is a degree of freedom in the definition of P, a plane in $\mathbb{R}^n$, when defined by $[w \quad \gamma] \in \mathbb{R}^{n+1}$.

[11] The equivalent expression in terms of $[w \quad \gamma]$ would be $\frac{\gamma}{\|w\|_2}$.

[12] The projections of the points to the plane are computed using the standard trigonometric functions.

- compute $distA$ and $distB$ /* vectors of projections to ray α */
- sort $distA$ and $distB$
- $CurrentMissed \leftarrow$ all points in $\mathcal{B}$
- $BestMissed \leftarrow CurrentMissed$
- $Position \leftarrow \min(\min(distA), \min(distB))$ /* initial position of plane */
- $BestPosition \leftarrow Position$
- REPEAT
  - move $Position$ to next point
  - update $CurrentMissed$
  - IF ($CurrentMissed < BestMissed$) THEN
    - $BestMissed \leftarrow CurrentMissed$
    - $BestPosition \leftarrow Position$
  - ENDIF
  - UNTIL (all points considered)
- return $BestMissed$ and store $BestPosition$

Figure 1: Pseudocode for the COUNTMISSED(α) oracle.

Sorting the distance vector is theoretically the bottleneck step in the oracle function. We use the C++ standard library function *sort* for this task[13]. The distances for A and B are stored and sorted separately to speed this step up.

3.2 Variable Neighborhood Search

A crucial insight underlying VNS is the observation that in many practical optimization problems with multiple local minima, these tend to be somehow clustered or correlated. Under these circumstances, a local minimum might contain information useful to find other, perhaps better, minima. The general idea in VNS is thus to try to escape from local minima by first exploring subsets of the domain that are in some sense "close" to the incumbent, and then going on to test for solutions that are increasingly different to it.

[13]Modern implementations of the C++ standard library use *introsort*, which has a worst case $O(n \log n)$ complexity, but is known to do much better on average [Mus97].

The only two exogenous ingredients for the basic version of VNS are a metric defined on the solution domain and a local descent procedure. The metric is used to build a problem-dependent structure consisting of K ordered neighborhoods, $\mathcal{N}_1(\alpha^*), \mathcal{N}_2(\alpha^*), ..., \mathcal{N}_K(\alpha^*)$ centered around the current incumbent solution point α^*. The first neighborhood $\mathcal{N}_1(\alpha^*)$ includes only solutions near the incumbent and subsequent neighborhoods include regions farther from it.

A local descent procedure is applied starting from a randomly chosen point within $\mathcal{N}_1(\alpha^*)$, and then from increasingly farther neighborhoods until a better solution is found or the stopping criterion is met. If an improved solution is found, the whole structure is recentered around it and the process restarts.

For the search over $\mathbb{R}^{n-1}$ in our problem, we adopt the following neighborhood structure:

$$\mathcal{N}_1(\alpha^*) = \prod_{i=1}^{n-1} \left[\alpha_i^* - \frac{1}{2K}, \alpha_i^* + \frac{1}{2K} \right]$$

$$\mathcal{N}_j = \left\{ \prod_{i=1}^{n-1} \left[\alpha_i^* - \frac{j}{2K}, \alpha_i^* + \frac{j}{2K} \right] \right\} \setminus \bigcup_{l=1}^{j-1} \mathcal{N}_l(\alpha^*) \quad \text{for } j = 2, ..., K$$

where $\prod$ denotes Cartesian product. A unit hyperbox is thus centered on α and sliced into K successive symmetric layers with equal thickness. Note that, the $\mathcal{N}_j(\alpha^*)$ being mutually exclusive, they do not correspond to topological neighborhoods[14].

Also note that the volume of each $\mathcal{N}_j(\alpha^*)$ is increasing in j. Since at each iteration one point for restarting the local search is drawn at random from each neighborhood, the implication is that the search is relatively more intensive near the incumbent. Figure 2 summarizes the algorithm.

The input to the local descent function is the starting point α and the output is $\widehat{\alpha}$, the new minimum found. $COUNTMISSED(\widehat{\alpha})$, the objective function evaluated at this point, is thus the number points misclassified by the best possible plane in direction $\widehat{\alpha}$.

For the local descent during the optimization phase, we use the downhill simplex method of [NM65]. Our implementation is roughly based on the structure suggested in [PFTV88].

The first neighborhood structure is arbitrarily centered at the origin. We use as stopping criterion for VNS the number of runs through all neighborhoods without improvement[15].

[14] $\mathcal{N}_1$, however, would be a neighborhood under the L_1-norm, in the topological sense of the word.

[15] Unless otherwise indicated, we set the limit of passes without improvement at 10, i.e. the algorithm stops if it has visited all neighborhoods ten times without improvement. This yielded reasonable results in moderate time in most of our test problems.

- Get initial point α^*
- $best \leftarrow \infty$
- REPEAT
  - $k \leftarrow 0$
  - REPEAT
    - Get random candidate $\alpha \in N_k(\alpha^*)$
    - Call local search; returns new local min
    - $new = COUNTMISSED(\hat{\alpha})$
    - IF $(new < best)$
      - $best \leftarrow new$
      - $\alpha^* \leftarrow \hat{\alpha}$
      - $k \leftarrow 0$
    - ENDIF
    - ELSE $k \leftarrow k + 1$
  - UNTIL $(k = K)$
- UNTIL (stop criterion met)
- Return α^* and $best$

Figure 2: Pseudocode for the VNS heuristic.

We now present the results of our numerical experiments.

4 Numerical Experiments

We try our exact formulation and our heuristic on a small set of problems from the famous UCI Machine Learning repository [BM98], and on a set of random problems created with Musicant's NDC, a publicly available generator [Mus98]. Preprocessing details and parameters used for the datasets are given in annex A.

Our exact solutions were obtained using the CPLEX callable library [ILO03], running under Linux[16]. Some of the problems we originally considered, such as the *Pima* database, failed to solve within the memory constraints of our solution setup[17].

We first compare the solution time of the LW model with our improved formulation (LWtight). The behavior of the heuristic is considered next: we study the full set fit and generalization properties of the solutions found by VNS by comparing it to known exact solutions, and then explore the solution time for sets of instances too large to be realistically tackled by an exact method with currently available technologies. Finally, we discuss the acceleration of exact solutions using our heuristic bound.

4.1 Exact Solutions

Table 1 compares the solution times of the original LW formulation with our improved version. The first column shows the objective value; the size of the problem is indicated in the next two columns, followed by the percentage of (full set) misclassified points at the optimum. In problems that take under a couple of seconds to solve, CPU time is not consistently accurate and comparisons should be made with caution. As the problems with substantial solution times, *Iris* took the same time under both formulations, and the tighter formulation of *Cancer* was solved in just 55% of the benchmark time. Much to our surprise, the *Housing* problem took longer to solve with the tighter formulation. We suspect this to be due to lengthier solutions for the LP relaxations at each node under the tighter formulation; this can happen, for instance, if a barrier method is used for the relaxations and the tighter MIP formulations result in their being nearly degenerate[18]. This phenomenon shows that the proposed formulation is not always better under the default settings, and that care should be taken in the choice of algorithms for the relaxations. We believe this issue to deserve further analysis.

Table 2 shows the corresponding results for the five random problems with 300 observations in 6 dimensions. In this case as well, the savings seem quite

[16]The processors are Intel Xeon 3.06 GHz, 1 Mb cache memory and 2 GO RAM.

[17]CPLEX appears to use some sophisticated memory management tricks [ILO03] and we did not explore alternative approaches. For comparability and to focus on the differences between the formulations, we used CPLEX's own branching scheme with default settings.

[18]In the sense of not having proper interior solutions.

					CPU seconds		
Problem	obj	obs	dim	bad rate	LW	LWtight	savings
cancer	13	683	9	1.9%	634.13	342.77	46%
echocardiogram	7	74	7	9.5%	2.31	2.31	0%
glass_windows	3	214	9	1.4%	1.31	1.20	8%
hepatitis	2	150	16	1.3%	1.77	1.78	-1%
housing	9	506	13	1.8%	6476.58	7499.78	-16%
iris	25	150	4	16.7%	54.25	54.17	0%

Table 1: CPU times for LW and LWtight formulations: UCI problems

significant. Although this sample is too small for a definitive assessment, it would seem that there is a correlation between the difficulty of the underlying classification problem, as estimated by the bad rate, and the solution time of the corresponding MIP.

				CPU seconds		
obj	obs	dim	bad rate	LW	tight	savings
21	300	6	7.0%	910.53	383.85	58%
30	300	6	10.0%	19011.81	15001.50	21%
21	300	6	7.0%	867.13	507.46	41%
6	300	6	2.0%	3.06	1.77	42%
23	300	6	7.7%	2549.89	1725.72	32%

Table 2: CPU times for LW and LWtight formulations: NDC problems

4.2 Heuristic Solutions

We first consider the speed and accuracy of the heuristic on the full sets (as opposed to training-testing partitions) of our UCI problems. Table 3 shows the best objective value found by the heuristic (VNS obj) and the implied fit, as well as the percentage of the time taken by it with respect to the exact solution; the exact solution and its accuracy are shown for reference.

The heuristic found the exact solutions in only two of the problems, but it did so quite fast. The gap in the objective value for the other cases is substantial. On the other hand, these approximations are obtained, for large problems, in a small fraction of the exact solution time. Consider, for instance, the *Housing* problem, with priors of about $\frac{1}{2}$, i.e. about half of the observations in each class. A plane that leaves about 91% of them on the correct side is obtained in .4% of the time it takes the exact algorithm to reach an accuracy of about 98%. Whether the trade off is worthwhile will depend

Problem	obs	dim	exact obj	fit	VNS CPU secs	fraction of exact time	VNS obj	fit
cancer	683	9	13	98.10%	16.51	4.8%	13	98.10%
echocardiogram	74	7	7	90.54%	0.88	38.0%	11	85.14%
glass_windows	214	9	3	98.60%	5.06	420.3%	6	97.20%
hepatitis	150	16	2	98.67%	6.47	363.3%	12	92.00%
housing	506	13	9	98.22%	29.72	0.4%	47	90.71%
iris	150	4	25	83.33%	5.21	9.6%	25	83.33%

Table 3: Full set accuracy and VNS running times: NDC problems

on the specific application, but we argue that the observed performance of the heuristic would be advantageous in many practical circumstances.

The huge difference in running times for the *Glass* and the *Hepatitis* problems (with VNS taking about four times longer than the exact method) should perhaps not be surprising. These are very easy instances, with exact solutions taking only a couple of CPU seconds; there is clearly no advantage in using any heuristic on them.

A key consideration when assessing a classifier is its off-sample performance. We performed a ten-fold cross validation exercise on the UCI datasets, comparing the average testing set accuracy of both the exact and heuristic solutions[19]. In order to have a reference to another, mathematical programming based linear discrimination alternative, we also compared them with the results obtained with RLP, a popular, double-plane linear programming discrimination model proposed in [BM92]. The results are summarized in figure 3. The heuristic generalized better than the exact solution on two problems, and worse on the other three. The performance of the exact solution on *Hepatitis* and the *Echocardiogram* problems was dramatically superior to both the heuristic and RLP[20]. These are, however, easy instances for which no heuristic is likely to be competitive in practice.

Studying the solution times of the heuristic on increasingly large datasets, with a fixed stopping criterion, we find that it scales up gracefully to problem sizes where exact solution is not an option. We created a hundred random problems in 6 dimensions with observations ranging from 2000 to 20000 by steps of 2000. Figure 4 shows the average running time of the ten random problems of each size, for a constant stopping criterion. On this range of

[19]For a detailed discussion of the k-cross validation and its properties, see e.g. [Han97, DHS00].

[20]It should be noted that RLP, unlike the other two methods being compared, does not attempt directly to minimize misclassifications on the training sets. However, the ultimate goal of any discrimination procedure being usually off-sample prediction, the comparison presented is relevant.

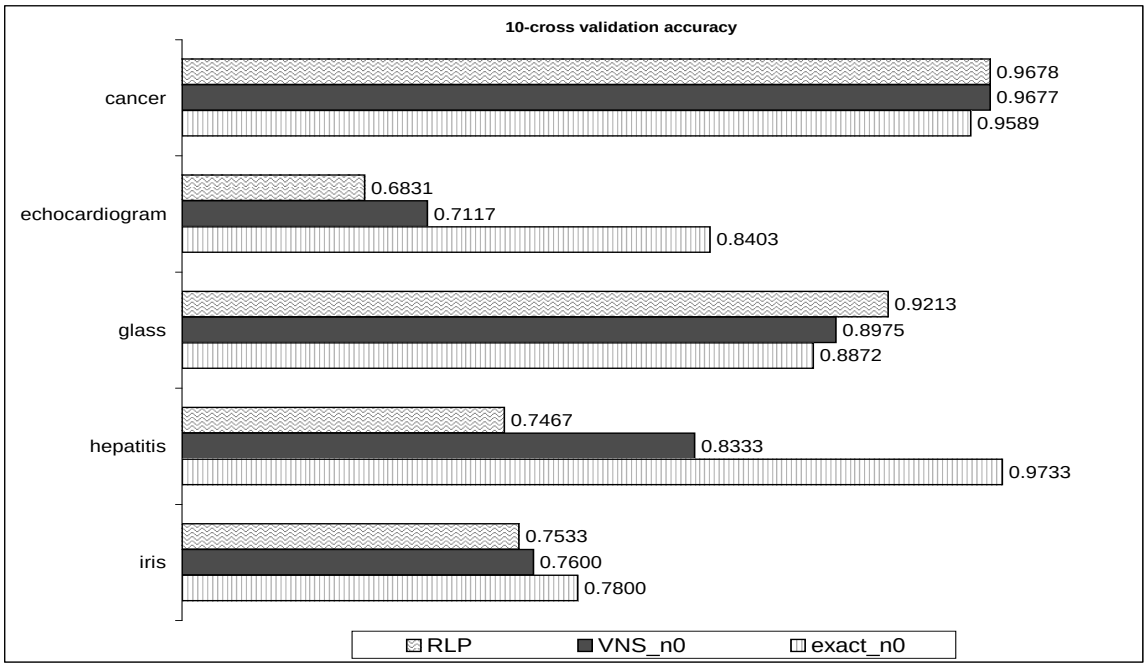

Figure 3: Generalization accuracy for alternative linear discriminants

problem sizes solution time appears to grow linearly.

An additional series of ten problems with 100000 observations each was solved in an average of 16 CPU minutes. Exact solution of these kind of problem is currently impossible. Although nothing can be said of the precision of our heuristic solutions, we have no reason to believe that the performance would be much worse than in the smaller problems for which an exact benchmark could be obtained, as discussed above.

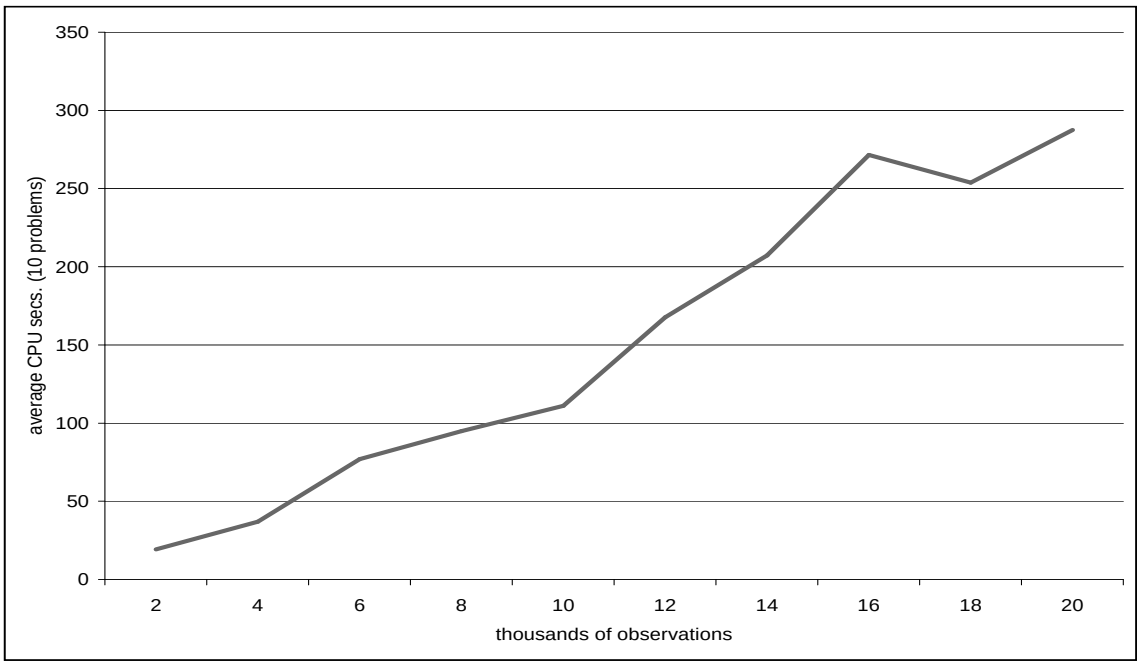

Figure 4: VNS solution time for NDC random problems in 6 dimensions.

4.3 Acceleration by Heuristic Bound

PENDING THE SOLUTION OF THE PROBLEMS WITH CPLEX

5 Conclusion

We review the main approaches to the misclassification minimization problem, and present and test an improved exact formulation. We also propose a heuristic based on the Variable Neighborhood Search framework.

Our tests suggest that the proposed exact formulation generally outperforms the classic benchmark on which it is based. Its use can be explored along with some of the algorithmic improvements introduced in the literature for this family of Mixed Integer Programs.

The heuristic appears to find reasonable solutions in small running times, and to perform acceptably in cross validation exercises. It scales up gracefully to problem sizes beyond the reach of exact methods.

We believe our approach to be merit further research.

A Data Set Details

We considered the instances from the UCI Machine Learning Repository [BM98] which either have only two classes or could be readily converted into a binary classification problem. We then retained those with very few or no categorical variables.

Cancer refers to the Wisconsin Breast Cancer database. Rows with missing attributes were deleted.

For the *Echocardiogram* problem, all instances with missing labels were deleted, and missing attributes were replaced by the corresponding class means.

For the *Glass* database the two classes considered were window versus non-window sources.

Housing refers to the Boston Housing database.

In the *Hepatitis* database, all observations with more than 6 missing attributes were deleted, as were columns 16 and 18, which had too many missing entries. Missing observations were then replaced with column means (if continuous) or modes. Column 3 trivially separates the set, and was also removed.

The problem known in the literature as *Pima*, the Pima Indians Diabetes database, with 768 observations in 8 dimensions, failed to solve exactly even with the best heuristic bound of .

Musicant's NDC generator is a matlab program. It locates randomly a given number of centers, assigns them to one of two classes by splitting the set with a randomly generated plane, and then produces multivariate normal observations from these centers, using a randomly generated covariance matrix. This approach provides some more generality than one might have with other common practices. Note, however, that even within the class of normally distributed problems, some reasonable, interesting configurations (such as having a small cluster centered on the "wrong" side of the plane) are not spanned by NDC. These limitations are inevitable in any exercise with artificial data, and we feel that replicability is facilitated with the use of a publicly available generator. The number of centers was made to be equal to the dimension of the problem and the dispersion parameter $nExpandFactor$ was fixed at 15.

All databases were linearly standardized to the range $[0, 1]$.

The original files are available at http:XXXXX SAY WHERE

References

[BCK99] E.K. Burke, P. Cowling, and R. Keuthen. New local and variable neighborhood search heuristics for a sequencing problem in printed circuit board assembly applied to the travelling sales-

man problem. Technical report, University of Nottingham, 1999.

[BH82] Steve M. Bajgier and Arthur V. Hill. An experimental comparison of statistical and linear programming approaches to the discriminant problem. *Decision Sciences*, 13(4):604, 1982.

[BM92] K Bennett and O Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[BM96] J. Brimberg and N. Mladenović. A variable neighborhood algorithm for solving the continuous location-allocation problem. *Studies in Locational Analysis*, 10:1–12, 1996.

[BM98] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[CH97] G. Caporossi and P. Hansen. Variable neighborhood search for extremal graphs 1: The autographix system. *Les Cahiers du GERAD*, G-97-41, 1997. Also in: Discrete Mathematics 212:29-44, 2000.

[CIS89] T. M. Cavalier, J. P. Ignizio, and A. L. Soyster. Discriminant analysis via mathematical programming: Certain problems and their causes. *Computers & Operations Research*, 16(4):353–362, 1989.

[CM96] C. H. Chen and O. L. Mangasarian. Hybrid misclassification minimization. *Advances in Computational Mathematics*, 5(2-3):127–136, 1996.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2000.

[FG86] Ned Freed and Fred Glover. Resolving certain difficulties and improving the classification power of lp discriminant analysis formulations. *Decision Sciences*, 17(4):589, 1986.

[Geh86] William V. Gehrlein. General mathematical programming formulations for the statistical classification problem. *Operations Research Letters*, 5(6):299–304, 1986.

[GKD88] Fred Glover, Sam Keene, and Bob Duea. A new class of models for the discriminant problem. *Decision Sciences*, 19(2):269, 1988.

[Gle99] J. J. Glen. Integer programming methods for normalisation and variable selection in mathematical programming discriminant analysis models. *Journal of the Operational Research Society*, 50(10):1043–1053, 1999.

[Glo90] Fred Glover. Improved linear programming models for discriminant analysis. *Decision Sciences*, 21(4):771, 1990.

[Gri72] Richard Grinold. Mathematical programming methods of pattern classification. *Management Science (pre-1986)*, 3:272–290, 1972.

[Han81] D. J. Hand. *Discrimination and Classification.* John Wiley & Sons, 1981.

[Han97] D. J. Hand. *Construction and assessment of classification rules.* Wiley, New York, 1997.

[HM97] P. Hansen and N. Mladenović. Variable neighborhood search for the p-median. *Location Science*, 5:207–226, 1997.

[HM98] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *Les Cahiers du GERAD*, G-98-20, May 1998. Published in: European Journal of Operational Research, 130:449-467, 2001.

[HMU00] P. Hansen, N. Mladenović, and D. Urosevic. Variable neighborhood search for the maximum clique problem. *Les Cahiers du GERAD*, G-2000-45, 2000.

[ILO03] ILOG, S.A. *ILOG CPLEX Callable Library 9.0 Reference Manual*, 2003.

[KE90] Gary J. Koehler and S. Selcuk Erenguc. Minimizing misclassifications in linear discriminant analys. *Decision Sciences*, 21(1):63, 1990.

[Ken61] M.G. Kendall. *A Course in the Geometry of n Dimensions.* Charles Griffin, London, 1961.

[Koe89a] Gary J. Koehler. Characterization of unacceptable solutions in lp discrimina. *Decision Sciences*, 20(2):239, 1989.

[Koe89b] Gary J. Koehler. Unacceptable solutions and the hybrid discriminant model. *Decision Sciences*, 20(4):844, 1989.

[Koe90] Gary J. Koehler. Considerations for mathematical programming models in discriminant analysis. *Managerial and Decision Economics*, 11(4):227–234, 1990.

[Koe91] Gary J. Koehler. Improper linear discriminant classifiers. *European Journal of Operational Research*, 50(2):188–198, 1991.

[LW78] J M Liittschwager and C Wang. Integer programming solution of a classification problem. *Management Science (pre-1986)*, 24(14):1515–1525, 1978.

[Man99] O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24(1-2):15–23, 1999.

[MM85] Edward P. Markowski and Carol A. Markowski. Some difficulties and improvements in applying linear programming formulations to the discriminant problem. *Decision Sciences*, 16(3):237, 1985.

[MMS95] P. Marcotte, G. Marquis, and G. Savard. A new implicit enumeration scheme for the discriminant-analysis problem. *Computers & Operations Research*, 22(6):625–639, 1995.

[MPKVC00] N. Mladenović, J. Petrovic, V. Kovacevic-Vujcic, and M. Cangalovic. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *Les Cahiers du GERAD*, G-2000-60, 2000. appeared in EJOR 151, 389-399.

[MS92] P. Marcotte and G. Savard. Novel approaches to the discrimination problem. *Zeitschrift für Operations Research (Theory)*, 36:517–545, 1992.

[Mus97] David R. Musser. Introspective sorting and selection algorithms. *Software Practice and Experience*, 27(8):983–993, 1997.

[Mus98] D. R. Musicant. NDC: normally distributed clustered datasets, 1998. www.cs.wisc.edu/dmi/svm/ndc/.

[NM65] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[PFTV88] William H. Press, Brian Flannery, Saul Teukolosky, and William Vetterling. *Numerical recipes in C : the art of scientific computing*. Cambridge University Press, Cambridge, 1988.

[PWL97] R. Pavur, P. Wanarat, and C. Loucopoulos. Examination of the classificatory performance of mip models with secondary goals for the two-group discriminant problem. *Annals of Operations Research*, 74:173–189, 1997.

[Rub90] Paul A. Rubin. Heuristic solution procedures for a mixed-integer programming discriminant model. *Managerial and Decision Economics*, 11(4):255–266, 1990.

[Rub97] P. Rubin. Solving mixed integer classification problems by decomposition. *Annals of Operations Research*, 74:51–64, 1997.

[Smi68] Fred W Smith. Pattern classifier design by linear programming. *IEEE Transactions on Computers*, C-17:367–372, 1968.

[Som58] D.M.Y. Sommerville. *An introduction to the geometry of n dimensions*. Dover Publications, New York, 1958.

[SS97] A. P. D. Silva and A. Stam. A mixed integer programming algorithm for minimizing the training sample misclassification cost in two-group classification. *Annals of Operations Research*, 74:129–157, 1997.

[Xia93] B. C. Xiao. Necessary and sufficient conditions of unacceptable solutions in lp discriminant-analysis. *Decision Sciences*, 24(3):699–712, 1993.