HEC Montréal
Affiliée à l'Université de Montréal

Essays on Linear Discrimination

par
Alejandro Karam

Option Méthodes Quantitatives de Gestion

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en Administration

Novembre, 2005

© Alejandro Karam, 2005.

Université de Montréal
Faculté des études supérieures

Cette thèse intitulée:

Essays on Linear Discrimination

présentée par:

Alejandro Karam

a été évaluée par un jury composé des personnes suivantes:

François Bellavance
président-rapporteur

Pierre Hansen
directeur de recherche

Michèle Breton
membre du jury

Jiming Peng
examinateur externe

Ann Langley
représentante du doyen de la FES

Thèse acceptée le 10 novembre 2005

RÉSUMÉ

Nous étudions la séparation de deux ensembles distincts de points dans un espace euclidien, par un hyperplan qui assigne un demi-espace à chacun des ensembles, en minimisant un certain critère d'erreur. Les essais dans la thèse traitent de plusieurs tels critères.

Le problème de la minimisation de la somme des distances L_p des points mal classés à l'hyperplan, pour une valeur de p vraiment arbitraire, entière ou fractionnaire, ne peut pas en général être résolue exactement. Nous appliquons à ce problème une heuristique basée sur la recherche à voisinage variable (*Variable Neighborhood Search*, VNS). Les solutions trouvées sont raisonnablement précises, et les temps de solution augmentent modestement avec la taille des problèmes. Nous constatons que le choix de p affecte les propriétés de généralisation du discriminant, mais d'une façon qui dépend du cas spécifique.

Nous utilisons les bornes obtenues par notre heuristique VNS pour accélérer la solution exacte dans les cas de $p = 2$ et $p = \infty$.

Nous considérons aussi le critère de la minimisation du nombre de points mal classés. Nous proposons une amélioration d'une formulation classique pour ce problème, et trouvons que, pour un certain nombre d'instances, elle réduit de manière significative le temps nécessaire pour obtenir une solution exacte. Nous implantons aussi le VNS pour ce problème, et constatons qu'il est rapide, et que les solutions qu'il trouve se généralisent raisonnablement bien par rapport à quelques autres classificateurs linéaires.

La dernière partie de la thèse est une application à la méthode des scores de crédit (*credit scoring*). Le critère de choix pour ce volet est la minimisation de la somme des déviations (i.e., pas nécessairement des distances) des points mal classés à l'hyperplan. Nous explorons la question de la sélection de variables dans le contexte d'un modèle de programmation linéaire. Notre travail est basé sur l'analyse empirique de deux bases de données réelles. Nous suggérons quelques améliorations à une méthode basée sur le principe du “jackknife”. Nous proposons également une réinterprétation de cette méthode, considérée comme traitant implicitement un problème bicritère. Notre interprétation suggère des formulations alternatives, et le potentiel d'une application plus ample des idées sous-jacentes à cette technique.

Mots clés: classification automatique, séparation en norme arbitraire, recherche à voisinage variable, exploitation des données, évaluation statistique du crédit, méthode des scores.

ABSTRACT

We consider the problem of separating two distinct sets of points in a Euclidean space with a hyperplane that assigns a half space to each of the sets, by minimizing some error criterion. The essays in the thesis deal with different such criteria.

The problem of minimizing the sum of L_p-norm distances of misclassified points to the plane, for truly arbitrary, integer or fractional, values of p, cannot in general be solved exactly. We successfully apply the Variable Neighborhood Search (VNS) metaheuristic framework to this problem. The solutions found are reasonably accurate, and scale gracefully to large instances. We find that the choice of p affects the generalization properties of the discriminant in a case-dependent manner.

Exact solution methods exist for the cases $p = 2$ and $p = \infty$. We use the VNS bounds to accelerate the exact solution for the last two of these cases.

The next criterion we consider is the minimization of the number of misclassified points. The exact solution of this problem leads to large MIP models that are often difficult to solve. We propose and test an improvement to a classic formulation, and conclude that, for a number of problems, it reduces significantly the exact solution time. We adapt and implement the VNS framework to find heuristic solutions for this, and find that it is fast, and that the solutions it finds generalize reasonably well as compared with some other linear classifiers.

The final part of the thesis is an application to credit scoring. It considers the criterion of minimizing deviations (which are not necessarily distances) of misclassified points to the plane, and explores the issue of feature selection in the context of a linear programming model. Our work is based on a case study of two real-life credit databases. We suggest and test some improvements to a method based on the jackknife principle. We also propose a reinterpretation of this method as implicitly dealing with a bicriterion problem. Our interpretation suggests alternative formulations, and a potential for wider application of the underlying ideas.

Keywords: automatic classification, arbitrary norm separation, feature selection, credit scoring, variable neighborhood search, misclassification minimization, data mining, machine learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

A mis padres Anuar y Nawel.

A mi Tonina.

A mi ahijado Matías.

ACKNOWLEDGMENTS

I have been fortunate to be directed by a marvelous research advisor. Pierre Hansen's wisdom and endless curiosity are at the same time a permanent challenge and a delight to both his students and his colleagues. His generosity and considerate guidance have marked me profoundly, and I will endeavor forever to remain worthy of his trust and friendship.

Michèle Breton is a source of awe and inspiration for those of us who have been privileged to live and work around her. Her constant support, manifested in so many ways over the years, has been decisive in this period of my life.

Even long before bringing me to HEC-Montréal, Georges Zaccour has been a crucial pillar of my academic and personal development.

I thank Jean-Marie Bourjolly for his support as member of my academic committee, and also Charles Audet, Francois Bellavance, Fabien Chauny, Genevieve Gauthier and Godfried Toussaint for the valuable insights obtained from discussions with them.

My beloved Eloísa has not written a word of this thesis and will probably not even read it. I acknowledge her, however, as a full coauthor of my work. Her laughter, her tears and her lovingly prepared soups impregnate these pages. Her being at my side has been the highlight of my years as a Ph.D. student and indeed of my life during most of the last ten years.

I am grateful to Patricia Armendariz and Carlos Noriega, whose mentorship continues to illuminate my career. I also appreciate Agustin Carstens's encouragement and support.

Many friends have helped me out of computer-related pitfalls (in Linux, C^{++}, CPLEX, LaTeX, MATLAB and similar horrors), and given me access to their resources. I'm especially indebted to (in alphabetical order) Heykel Achour, Michel Aka, Slim Belhaiza, Gilles Caporossi, Özgür Gürtuna, Khemais Hammami, Hicham Lahlou, Sébastien Le Digabel, Guiomar Martín-Herrán, Sylvain Perron and Walid Zghal.

I would not have survived (and so much enjoyed) my coursework without the help of Amira, Karim, Karima, Lotfi, Mounir, Ramla, Salma, Sihem, Sophie, Tarek, and most specially Eymen.

Substantial parts of chapter 2 have been incorporated in a paper prepared with C. Audet, P. Hansen, C. T. Ng and S. Perron. I gratefully acknowledge their comments and corrections, as well as those of an anonymous referee, which have of course been fed back to this thesis. Professor O. L. Mangasarian pointed me to the NDC data generator.

The administrative staff, both at my department (Management Science at HEC-Montréal) and at GERAD, has been very supportive of my roles as student, research assistant and lecturer. Many thanks to Any, Carmelle, Carole, Céline, Dianne, Francine, Lise and Lucie. I am also grateful for all the assistance received from, and miracles performed by Lise Cloutier-Delage at various stages of the doctoral program.

Several people helped me to obtain access to private databases. Others spent a lot of time, in person and on the phone, helping me make sense of them. In compliance with my confidentiality agreements I do not disclose their identities. But to them, and the many other friends who know they belong in this page, goes my heartfelt gratitude and recognition.

I thank Pollo for teaching me to use Excel macros for my Bachelor's thesis. Better (over a decade) late than never.

I gratefully acknowledge support from CONACYT's graduate loan program.

INTRODUCTION

For over half a century, linear discriminant functions have been applied by statisticians, computer scientists and operations researchers to a wide array of supervised classification problems. An optimization dimension is present in every model of classification, whether it is developed within a statistical, artificial intelligence or operations research mindset. There is, however, a branch of the literature that approaches the problem specifically from a mathematical programming perspective. This thesis belongs to that branch; it is a collection of essays, each presented in a chapter, on various aspects of the construction of linear discriminant functions (for brevity also called just "linear discriminants"), mostly from the perspective of the optimization challenges involved.

The specific problem we deal with can be summarized as follows. Suppose we have two distinct sets $\mathcal{A}$ and $\mathcal{B}$, called *classes*. The members of these sets are of the same kind as those that we wish to classify. We call them *observations* or *points*, and they can represent clients, tumor tissue samples, credit applicants, transactions, and so on[1]. For each of these points, we can observe a number (say n) of *features* or *characteristics*[2]. It is assumed that the observations in each class are essentially different in some sense relevant for the task at hand. The general supervised (binary) classification problem[3] is then: given a new observation (with its corresponding features), whose true class belonging is unknown, should we assign it to $\mathcal{A}$ or $\mathcal{B}$?

It is useful to visualize the sets $\mathcal{A}$ and $\mathcal{B}$ as points in n-dimensional Cartesian space $\mathbb{R}^n$. A linear discriminant can then be seen as a hyperplane in this space. The half-spaces defined by it are assigned to each of the two classes; a new observation will be predicted to belong to either class depending on which side of the hyperplane it appears. In most problems of practical interest, any plane will leave points from at least one of the two sets on both sides (i.e., the convex hulls of $\mathcal{A}$ and $\mathcal{B}$ intersect) and perfect discrimination is not possible. The challenge in the construction of a linear discriminant is to find a plane that is the best in some reasonable sense.

In mathematical programming approaches to classification, the hyperplane is found by minimizing some expression of the error with which it separates the

[1]In the literature they are also denominated *individuals* or *patterns*.

[2]Also called *measurements* or simply *variables* in statistics.

[3]Although many supervised classification methods can deal with multi-class problems, we concentrate on the two-class case.

classes. One such criterion is the sum of L_p-norm distances to the plane of misclassified points, which has some intuitive appeal but poses a substantial optimization challenge. For the cases of $p = 1$ (known as the Manhattan distance), $p = 2$ (the Euclidean distance) and $p = \infty$ (the max-norm distance), techniques are known to find exact solutions. For a truly arbitrary choice of p, perhaps even fractional values, no exact algorithm is available, and one has to resort to heuristics. Chapter 1 deals with this problem. We propose and explore a heuristic based on the Variable Neighborhood Search framework.

A crucial consideration for the choice of a criterion to construct the discriminating hyperplane is how well the resulting rule performs in correctly classifying new observations (as opposed to looking only at how it classifies the observations on which it is built). An interesting question in this context is how the choice of the L_p-norm (i.e., the value of p) affects the discriminant's predictive accuracy. This issue is also discussed in chapter 1 where we study the generalization properties of discriminants obtained under different norms on a set of real-life instances from the famous UCI repository.

Chapter 2 considers the three cases for which an exact solution can be computed, namely the norms 1, 2 and infinity. It has been established in the literature that the case of the L_1-norm can be solved with $2n$ linear programs. The L_2-norm case can be formulated as a quadratic program with non-convex quadratic constraints, and is tackled in Sylvain Perron's recent doctoral dissertation from the École Polytechnique de Montréal. Finally, for the L_∞-norm case, we study a Mixed Integer Programming (MIP) formulation. We discuss the predictive accuracy obtained with these three criteria, again on the UCI examples. In this essay we also show how our heuristic solution, exposed in chapter 1, can be used to accelerate exact algorithms.

Other than measuring the L_p-norm distance of misclassified points to the plane, an alternative way of characterizing the efficacy of a linear discriminant is to consider the number of misclassified points[4]. In fact, this is the criterion under which any discriminant is ultimately assessed. However, minimizing it directly is a difficult task, normally modeled with Mixed Integer Programs in which, in practice, only small instances can be solved exactly. A distinct vein of the literature has been devoted to both exact and heuristic approaches for this problem. In chapter

[4]Curiously, because of a limit consideration on the definition, this criterion is sometimes referred to as the L_0-norm.

3, we review this literature, propose a formulation that improves a classic model, and adapt and apply our VNS heuristic to the setting of this problem of misclassification minimization.

Linear discriminants are used in practice by comparing a weighted sum of the features with some threshold value; the weights and the threshold determine the discriminant and characterize the hyperplane in the geometrical interpretation. The weighted sum in question is called in some applications a *score*. Rather than considering the actual distance of points to the plane or the number of misclassified points, as in the models mentioned above, another possible criterion for constructing a linear discriminant is to minimize some measure of deviation of the scores of misclassified points with respect to some reference value (related to the threshold). This approach has the advantage of fully preserving the linearity of the objective function, and is taken in a prolific vein of the literature. We explore it in the context of an application to credit scoring in the last part of the thesis.

While the literature on discrimination for corporate bankruptcy prediction has been well established for decades, it is only in recent years that the specific challenges of consumer credit scoring have received (dramatically increasing) attention from academia, as results from research with proprietary data are published, with appropriate reserves. Chapter 4 presents a brief review of issues in the construction and use of linear discriminants for credit scoring, while in chapter 5 we review the literature on linear programming (LP) approaches to discrimination and test alternative formulations for this problem. We also study the crucial problem of feature selection in the context of LP based discrimination, and propose a reinterpretation of a classic method. We were fortunate to have access to two large, real life credit databases, one of mortgages and the other of car loans, on which our empirical work is based.

The chapters are linked by a thematic thread and have some cross references to each other. They are, however, conceived to be read as self contained essays. Therefore, considerable redundancy is inevitable, and the valiant reader of the entire thesis is asked for patience and forgiveness for some repeated definitions and arguments along successive chapters.

A brief general conclusion follows the essays, and a common appendix describes in detail the databases used in chapters 1, 2 and 3.

NOTATION

The following notation will be used across the chapters.

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.

$\mathbb{R}^n_+$ is the closed positive orthant.

Following [Man99], we denote $\mathcal{A}$ and $\mathcal{B}$ the two sets of points in $\mathbb{R}^n$ containing respectively m and k points and represented by the matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{k\times n}$.

For a given plane

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

a point $x \in \mathcal{A} \cup \mathcal{B}$ is said to be misclassified when

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

This is equivalent to the convention that the gradient w points to the half space corresponding to $\mathcal{A}$.

CHAPTER 1

ARBITRARY-NORM SEPARATION BY VARIABLE NEIGHBORHOOD SEARCH

1.1 Introduction

Since Fisher's seminal work [Fis36], linear discriminant functions have been fundamental tools for automatic classification. Once the parameters of such models are defined, new observations can be classified with great ease and speed. This has made them, despite their limitations, popular for a wide variety of applications. Furthermore, by expansion or transformation of the original observation space, these techniques can be applied to non-linear discrimination.

The linear discriminant can be seen as a hyperplane that somehow separates the representations of the training observations of each of the two classes in an n-dimensional real space $\mathbb{R}^n$. The coefficients of the plane will provide the desired discriminant.

When the respective convex hulls of the two sets of observations intersect, perfect separation with the plane is impossible. As some points will necessarily lie on the wrong side of any plane, a natural idea is to choose the separating plane by minimizing some measure of the aggregate error implied by such misclassified points.

An intuitively appealing measure of this error is the sum of the distances to the plane of the misclassified points. As this distance is, in general, non-linear, the resulting optimization problem is quite difficult. Alternative approaches have therefore been used, that preserve the linearity of the objective function, but in fact renounce the precise distance as the criterion. This is often done by minimizing some other measure of deviation of misclassified points with respect to the plane [Smi68, Gri72, FG81a, Han81, LO90].

A recent vein in the literature has emerged that tackles explicitly the problem of minimizing the sum of distances to the plane. Analytical expressions for the distance in arbitrary L_p-norm from a point to a plane are derived in [Mel97], but [Man99] appears to be the first to apply them to the separating hyperplane problem[1]. He shows that the L_1-norm case can be solved with $2n$ linear programs

[1]Mangasarian also presents his own proof of the formulae. Another, perhaps simpler proof is

(each with about as many variables and as many constraints as there are training set observations), and suggests a formulation with a bilinear objective function and convex constraints for the L_2-norm case. The implications of these precise formulations for data mining applications are yet to be fully explored, and it is in this context that we consider our work.

In [AHK$^+$04] a linear mixed integer formulation is presented for the L_∞-norm, and a branch and cut approach [AHJS00, Per04] is applied to Mangasarian's formulation of the L_2-norm case.

These developments have been confined to the more usual cases of the L_1, L_2 and L_∞-norms, and except to some extent for the L_1-norm, are in practice unsuitable for application to large scale problems.

We here present a heuristic method that can be applied to a truly arbitrary (integer or fractional) norm, and that can scale up gracefully to relatively large instances. The general idea is to project the objective function (i.e., the sum of distances in the given L_p-norm, of misclassified points to the plane) to the surface of a Euclidean unit sphere representing the possible directions of the gradient of the plane. The heuristic search for a minimum is then performed over this sphere.

The two key insights behind our approach are the following:

- The search for an optimal hyperplane can be decomposed in determining its direction and then, for a given direction, its position or offset with respect to the origin;
- Once a direction is fixed, the conversion from the Euclidean distance to an arbitrary L_p-norm is just a constant rescaling.

The procedure is implemented in two steps. We first construct a query function such that, given any point on the unit Euclidean sphere (i.e., any direction for the gradient of the candidate plane), determines the optimal position of the plane along that direction, and computes the sum of distances (in the given L_p-norm) of misclassified points to the best plane lying in that direction. In the second step, the optimization phase, the heuristic takes this function as an oracle and queries it repeatedly as it searches the surface of the sphere for the minimum.

Our implementation of the optimization phase uses Variable Neighborhood Search (VNS) [HM01b], an approach which has proved to be useful in a vari-

given in [PC01].

ety of difficult global optimization problems [BM96, CH00, HM97, BCK99, HMU04, MJPČ03].

Our numerical tests suggest that this algorithm often finds optimal or near optimal solutions. As shown in chapter 2, these heuristic solutions can also be used to significantly accelerate exact methods.

Since the proposed approach works for truly arbitrary norms (other than the traditional 1, 2 and ∞), we can explore for the first time the effects of the choice of p on the generalization properties of L_p-norm hyperplane separation.

The rest of the chapter is organized as follows. Section 1.2 establishes the notation and explains the query function to be used as oracle in the search process. Section 1.3 summarizes the VNS approach and presents the search phase. Section 1.4 then summarizes our numerical results. The final section concludes.

1.2 The Oracle

Recall that $\mathcal{A}$ and $\mathcal{B}$ denote the two sets of points in $\mathbb{R}^n$ containing respectively m and k points represented by the matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{k\times n}$

The discriminating hyperplane $P = \{x \,|w^t x = \gamma\}$ can also be characterized by a set of $n - 1$ angles, $\alpha = (\alpha_1, \alpha_2, ..., \alpha_{n-1})$ which determine a direction perpendicular to the plane, and by an offset φ from the origin along this direction. The conversion from (α,φ) to the corresponding (w, γ) can be performed by an iterative procedure [Ken61] [Som58].

Once the training data set $\mathcal{A} \cup \mathcal{B}$ and the desired norm p are fixed, the input to the oracle is a direction α from the origin. We adopt the convention that the set $\mathcal{A}$ is meant to lie on the side of the plane to which w points.

The output PNORMDIST(α) is the sum of p-distances of misclassified points by the best possible plane along the given direction.

The algorithm is outlined in Figure 1.1.

The points in $\mathcal{A}$ and $\mathcal{B}$ are projected into the ray defined by α. Their distances to the origin are recorded and ranked (distances to the origin are signed, a negative value indicating the direction opposite to the origin with respect to w).

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 sum of distances of misclassified points

```
• $missA \leftarrow 0$        /* counter of missed points from $\mathcal{A}$ */
• $missB \leftarrow |\mathcal{B}|$        /* counter of missed points from $\mathcal{B}$ */
• compute $distA$ and $distB$ /* vectors of projections to ray $\alpha$ */
• sort $distA$ and $distB$
• $sumdist \leftarrow$ sum of all entries in $distB$
• $bestdist \leftarrow sumdist$
• $bestpoint \leftarrow$ first point
• $current \leftarrow \min(\min(distA), \min(distB))$        /* initial position of plane */
• REPEAT
    • move $current$ to next point
    • update $missA$ , $missB$ and $sumdist$
    • IF $(sumdist < bestdist)$
        • $bestdist \leftarrow sumdist$
        • $bestpoint \leftarrow current$
      ENDIF
  UNTIL $current = \max(\max(distA), \max(distB))$ /* all points considered */
• $pbest \leftarrow bestdist/DualNorm$
• return $pbest$
```

Figure 1.1: Pseudocode for the $PNORMDIST(\alpha)$ oracle.

$sumdist$ corresponds to the distance of all points in $\mathcal{B}$ to the initial plane.

The plane is then moved along the ray, in the direction of α, to the position corresponding to each successive point. The distance between successive projections and the updated counters $missedA$ and $missedB$ are used to compute the new $sumdist$. The minimum distance $best$ found along the way is recorded. The position where this minimum occurs is also noted.

By equation (7) in [Man99], the conversion of this value to the corresponding sum of distances in p-norm is obtained by dividing it by the dual norm $\|w\|_p'$ of the gradient w that corresponds to the given α. For $1 < p < \infty$, $\|w\|_p' = \|w\|_q$ where q is such that $\frac{1}{p} + \frac{1}{q} = 1$. We therefore compute it as

$$\|w\|_p' = \sqrt[\frac{p-1}{p}]{\sum w^{\frac{p}{p-1}}} \; .$$

We also recall that $\|w\|_1' = \|w\|_\infty$ and vice versa, and deal with these cases separately.

The adjusted value $p - best$ is then the minimum sum of L_p-norm distances of points misclassified by a plane in direction α. This value is returned by the oracle.

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

Sorting the distance vector is theoretically the bottleneck step in the oracle function. We use the C++ standard library function *sort* for this task[3]. The distances for $\mathcal{A}$ and $\mathcal{B}$ are stored and sorted separately to speed this step.

1.3 The Heuristic Search

Variable Neighborhood Search is a metaheuristic framework that has been successfully applied to a variety of difficult optimization problems. The basic methodology and several extensions can be found in [MH97], [HM99], [HM01b], and [HM01c]. Applications to unsupervised classification include [HM01a], [BHM02] and [HJM98].

In the realm of supervised classification, [GTMM03] use VNS to deal with combinatorial challenges arising in variants and refinements of the k-nearest neighbors approach.

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 from it.

Among the family of metaheuristic frameworks, VNS has a number of advantages that in our view make it suitable for the optimization phase of this data mining problem. It is simple and transparent, with a very small number of parameters; it is also quite flexible, allowing for any local search procedure and objective function to be used without changing the general framework.

The only two exogenous ingredients for the basic version of VNS are a metric

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

[3]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].

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 search 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:

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

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

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[4].

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 1.2 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. $PNORMDIST(\widehat{\alpha})$, the objective function evaluated at this point, is thus the sum of distances of misclassified points to the best plane in direction $\widehat{\alpha}$.

For the local descent during the optimization phase, we use the downhill simplex algorithm of [NM65]. This is a derivative-free method that explores the solution space by iteratively modifying an irregular polyhedron of $n + 1$ vertices. Our implementation is roughly based on the structure suggested in [PFTV88].

The first neighborhood structure is arbitrarily centered at the origin. We use as

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

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

Figure 1.2: Pseudocode for the VNS heuristic.

stopping criterion for VNS the number of runs through all neighborhoods without improvement[5].

We now present the results of our numerical experiments.

1.4 Experimental Results

We tested our algorithm on several series of random problems, as well as on a set of real life instances from the UCI Repository [DNM98]. For moderately sized instances, exact solutions can be obtained for p values of 1, 2 and ∞. We first compare our heuristic's performance to these exact results, and then explore the algorithms's speed on larger problems, for which exact benchmarks are impossible or impractical to obtain.

Although known exact methods can be applied to moderately sized problems in 2-norm and ∞-norm, and to some extent to larger problems for the 1-norm,

[5]Unless otherwise indicated, we set the limit of passes without improvement at 6 (i.e., the algorithm stops if it has visited all neighborhoods six times without improvement). This yielded reasonable results in moderate time in most of our test problems.

for the case of a truly arbitrary, integer or fractional $p > 1$, there are no available general exact methods. An interesting issue for practical data mining applications is whether a choice of p results in better generalization for a class of problems. Our heuristic approach allows this issue to be tackled. In the final part of this section, we study the effects of using a range of values for p on the generalization properties of the resulting planes, as estimated by k-cross validation on a set of UCI problems.

1.4.1 Benchmarks to Known Exact Solutions

For a set of real life instances from the UCI Repository and two sets of artificial problems, we compare the heuristic solutions found by our algorithm for the L_1-norm, L_2-norm and L_∞-norms to the corresponding exact solutions.

For the L_1-norm case, the exact solutions were found by solving $2n$ linear programs, as suggested in [Man99]. We solved these $2n$ LPs sequentially[6], but accelerated the solution process by adding at each step the constraint that the objective value be at most the best found so far. This technique can save a number of phase II solutions and reduces considerably total computation time.

The exact L_2-norm and L_∞-norm solutions were found with the techniques proposed in [AHK$^+$04].

The random problems were created with D. Musicant's NDC generator [Mus98], which produces normally distributed clusters. This generator is publicly available and has been used in other discrimination studies (e.g., [FM01, MM00]).

For the first series of random problems, we fixed the dimension at 6 and explored the effects of increasing the number of points from 2000 to 20000 (by steps of 2000). As the exact solution in L_1-norm is much easier than for the other norms, we were able to extend the series to 100000 by increments of 20000 for this case.

We then generated a second series of problems with 2000 points, with dimensions ranging from 4 to 13. For each problem size, we generated 10 instances and report the corresponding mean values of the results.

The parameters used for the generation of the artificial sets, as well as the preprocessing steps for the real life instances from the UCI Repository are detailed in the appendix.

Tables 1.1, 1.2 and 1.3 present, respectively, the results for the Irvine datasets and each of the two families of artificial problems. The gap columns shows the percentage above the exact minimum of the objective function value found by the

[6] As the $2n$ linear programs are independent, a parallel implementation would be possible.

heuristic. The fit column shows the full-set accuracy of the discriminant (i.e., the percentage of points correctly classified by the corresponding hyperplane). The computation time in seconds is shown in the CPU column. Results are shown for the L_1-norm, L_2-norm and L_∞-norms.

			L1-norm					L2-norm					L∞-norm				
			VNS			EXACT		VNS			EXACT		VNS			EXACT	
problem	dim	obs	gap %	fit	CPU	fit	CPU	gap %	fit	CPU	fit	CPU	gap %	fit	CPU	fit	CPU
cancer	9	683	0.00%	96.78%	5.3	97.07%	0.08	0.50%	97.07%	6.2	97.07%	156	0.01%	97.37%	7.6	97.51%	1.1
echocardiogram	6	74	0.00%	75.68%	0.5	77.03%	0.01	0.03%	75.68%	0.6	75.68%	5	0.00%	72.97%	0.5	77.42%	0.1
glass	9	214	2.33%	95.79%	3.3	95.79%	0.03	0.79%	94.86%	3.5	95.33%	5	0.82%	94.39%	2.8	95.79%	0.4
hepatitis	16	150	0.60%	88.00%	4.3	90.00%	0.06	1.17%	86.67%	4.4	88.00%	14,181	3.49%	89.33%	7.4	90.00%	65.7
housing	13	506	0.17%	85.77%	28.8	84.58%	0.14	0.57%	84.19%	13.9	84.39%	550	4.50%	86.76%	13.2	88.14%	30.4
pima	8	768	0.00%	72.79%	4.7	73.05%	0.42	0.00%	75.39%	4.4	75.39%	612	0.00%	76.04%	4.4	76.04%	8.1

Table 1.1: UCI instances.

Exact solution times are not reported for the random problems in L_2-norm for 12 and 13 dimensions, since these exact solutions could not be obtained within the predetermined maximum allowed CPU time without the heuristic bound[7].

As far as the objective function is concerned, the heuristic finds the exact solution in most of our datasets, and is close on many of the cases where it does not. As we can see from the results for *housing* and *hepatitis*, as well as the last lines of table 1.2, performance deteriorates on dimensions above about 10, but appears to be robust to the number of observations[8]. The *glass* problem, although only in 9 dimensions, appears to have a very large number of local minima under the L_1-norm, and the heuristic fails to find the optimum within the limits set by our stopping criterion. Interestingly, the full set fit of the heuristic plane is the same as that of the exact solution.

For the L_1-norm, the gap was zero in nine out of the 10 problems with 18000 observations in 6 dimensions, while the remaining one failed by almost 13%; the heuristic found the exact solution for all but one of the problems with 80000 observations, which was missed by an (abysmal) 32%.

The small but systematic apparent disadvantage of the heuristic solution in full set accuracy for some problem sets appears to be due to round-off and precision differences between our algorithm and the exact solver.

[7]The use of heuristic solutions to accelerate exact solution in L_2- and L_∞-norms is discussed in chapter 2.

[8]Poor performance on higher dimensions can be traced to the limitations of the local search technique, the Nelder-Mead simplex search.

For this range of problem sizes, the computing time for our heuristic appears to be linear in the number of observations, and quadratic in the dimension[9]. The time gains are most dramatic with respect to the L_2-norm, which is the hardest of the three to compute exactly. For the problems with 2000 observations in 11 dimensions, the heuristic found the optimum in about $\frac{1}{600}^{th}$ of the time of the exact algorithm. Exact solutions under the L_1-norm are of course the easiest to obtain, and the heuristic approach is not worthwhile for small problems (such as our Irvine examples and problems with up to 10000 observations). However, even under the L_1-norm, the heuristic looks advantageous in all instances larger than 20000 observations, and the difference reaches a full order of magnitude for 100000 points, with very little loss of average accuracy.

[9]Least squares fit of the model $CPUtime = \beta_0 + \beta_1 * dim + \beta_2 * dim^2$ yields an R^2 of .977 with $\beta_0 = 17.0918$, $\beta_1 = -5.09$ and $\beta_2 = 0.6314$.

		L1-norm					L2-norm					L∞-norm				
		VNS			EXACT		VNS			EXACT		VNS			EXACT	
dim	obs	gap %	fit	CPU	fit	CPU	gap %	fit	CPU	fit	CPU	gap %	fit	CPU	fit	CPU
4	2000	0.00%	94.33%	8.5	94.36%	0.4	0.00%	94.41%	10.5	94.41%	5	0.00%	94.29%	8.3	94.31%	0.8
5	2000	0.00%	93.68%	7.6	93.70%	0.5	0.00%	93.83%	11.1	93.84%	11	0.00%	93.69%	7.1	93.71%	1.5
6	2000	0.00%	92.95%	7.5	92.98%	0.6	0.00%	93.21%	13.9	93.23%	34	0.00%	93.11%	6.9	93.14%	3.9
7	2000	0.51%	90.98%	11.3	90.92%	1.0	0.00%	91.73%	21.8	91.75%	118	0.00%	91.49%	9.4	91.54%	8.8
8	2000	0.53%	89.29%	15.3	89.46%	1.4	0.00%	90.13%	29.2	90.16%	557	0.00%	89.63%	14.0	89.69%	20.6
9	2000	0.01%	89.85%	22.8	89.96%	1.6	0.00%	90.13%	44.3	90.17%	1,796	0.05%	89.92%	21.4	90.06%	49.4
10	2000	0.01%	88.86%	27.5	88.94%	1.9	0.00%	89.51%	56.8	89.55%	3,194	0.00%	89.37%	28.2	89.46%	102.8
11	2000	0.00%	82.32%	43.4	82.45%	3.4	0.00%	83.65%	64.0	83.74%	38,531	0.02%	83.48%	31.4	83.59%	330.9
12	2000	2.09%	87.61%	49.0	88.25%	2.8	0.01%	88.87%	78.5	89.02%	N/A	0.04%	88.41%	42.3	88.58%	530.5
13	2000	10.12%	84.49%	53.7	84.71%	4.0	0.01%	86.02%	93.8	86.15%	N/A	0.27%	85.56%	49.7	85.73%	1324.6

Table 1.2: Artificial problems with fixed number of observations.

Table 1.3: Artificial problems with fixed dimension.

		L1-norm					L2-norm					L∞-norm				
		VNS			EXACT		VNS			EXACT		VNS			EXACT	
dim	obs	gap %	fit	CPU	fit	CPU	gap %	fit	CPU	fit	CPU	gap %	fit	CPU	fit	CPU
6	2000	0.00%	93.12%	8.1	93.14%	0.7	0.00%	93.14%	14.6	93.15%	36	0.00%	93.10%	7.1	93.13%	4
6	4000	0.00%	92.16%	15.2	92.17%	3.0	0.00%	92.22%	28.3	92.22%	213	0.00%	91.90%	13.5	91.92%	15
6	6000	0.21%	91.08%	18.5	91.21%	8.9	0.00%	91.59%	34.2	91.60%	636	0.00%	91.51%	16.5	91.53%	41
6	8000	0.00%	91.65%	36.0	91.66%	16.1	0.00%	92.27%	54.9	92.28%	960	0.00%	92.24%	33.8	92.25%	65
6	10000	0.00%	91.22%	50.5	91.22%	27.3	0.00%	91.04%	77.7	91.04%	1,433	0.00%	91.08%	45.2	91.08%	126
6	12000	0.00%	90.70%	35.4	90.71%	44.2	0.00%	91.01%	69.6	91.01%	2,593	0.00%	90.94%	32.9	90.94%	204
6	14000	0.00%	89.34%	40.3	89.35%	62.6	0.00%	89.68%	74.9	89.69%	2,858	0.00%	89.56%	37.2	89.58%	321
6	16000	0.00%	92.09%	79.1	92.10%	54.3	0.00%	92.25%	112.4	92.25%	2,146	0.00%	92.21%	70.6	92.22%	288
6	18000	1.27%	92.57%	57.8	92.61%	77.5	0.00%	92.80%	115.5	92.80%	5,916	0.00%	92.82%	52.1	92.82%	391
6	20000	0.00%	94.48%	58.1	94.48%	96.7	0.00%	94.59%	118.5	94.59%	5,777	0.00%	94.49%	57.1	94.49%	368
6	40000	0.00%	92.18%	193.28	92.18%	528.4										
6	60000	0.00%	95.06%	162.25	95.06%	1000.7										
6	80000	3.23%	91.58%	233.16	91.84%	2039.2										
6	100000	0.00%	90.51%	306.83	90.51%	4384.7										

1.4.2 Test on Larger Problems

We explored empirically the computation time of our algorithm on larger problems, for which exact benchmarks are unavailable. In order to control for problem difficulty, we constructed an additional set of nine test problems, with a simpler structure than those used for benchmarking accuracy[10]. The dimension was fixed at 10, and the number of observations was increased from 200000 to one million, by steps of 100000.

Figure 1.3 shows the CPU time of the heuristic (for the Euclidean norm) as function of problem size. For this range of problem sizes, computation time still appears to grow linearly; the instance with one million points in ten dimensions took a little under two hours to run.

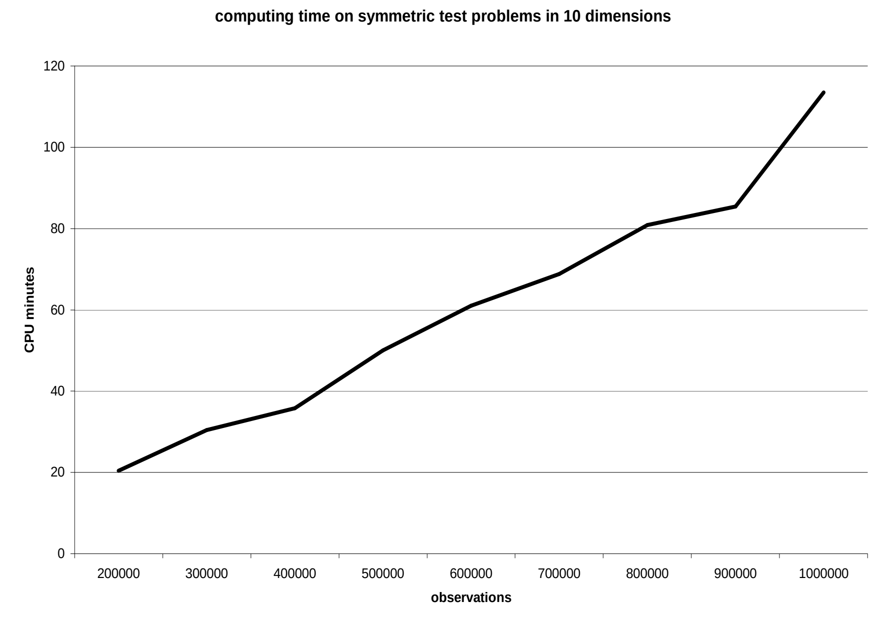

Figure 1.3: Running time for large problems.

We used for these experiments the same stopping criterion as in the accuracy benchmarks discussed in the previous section. However, for all of these large problems, the heuristic solution that's eventually reported is found very quickly (within the first 3% to 5% of the total time), the rest of the time being spent spanning the neighborhoods without further improvement. In our example with one million points, the final solution was found in only 4.5 minutes.

[10]The problems were constructed by fixing two arbitrary centers, assigning one half of the points to each and generating independent columns from a normal distribution for each of them.

We suspect this phenomenon to be due to the fact that in problems with simple structures, with a large number of points in moderate dimensions, the effect of each single point on the surface of the objective function is relatively small, and deep, narrow valleys are unlikely to exist. Under these circumstances, just a few runs of the local descent lead to the global minimum. This suggests that, when this kind of problem is identified, and complex clusters of local minima are not present, the use of our oracle with a suitable descent routine can provide a good, quick answer even in fairly large instances.

1.4.3 Classification Accuracy with Arbitrary Norm

We now turn to the study of the effect on classification accuracy of the choice of the norm p for the UCI datasets. Our experimental set up is as follows. We compute the 10-cross validation accuracy of separating hyperplanes that minimize the p-norm distance of misclassified points, as found by our heuristic, for 30 values of p, starting from 1 with increments of .2. In order to smooth out the effect of the partition, we repeat the process 10 times for each data set[11].

As accuracy was more important than speed for this exercise, the stopping criterion for VNS runs was raised to 10 passes through all neighborhoods without improvement. As we did not expect the optimal planes to vary dramatically for nearby values of p, the VNS search was initiated at the best point found for the previous value of p considered[12].

The results are plotted in figures 1.4 to 1.9. The lighter lines represent the 10-cross validation accuracies obtained with each partition; the darker line is the average. For *echocardiogram, glass* and *hepatitis*, there seems to be no systematic effect of the value of p on testing set classification accuracy. For *cancer*, despite some fluctuations, it appears that the planes with $p = 1$ perform generally at least as good as any others.

Results for the *housing* data set show an increasing trend on p, with average accuracy raising from 82% to 84% as p goes from 1.6 to 6.8. For the *pima* data set the effect is quite remarkable: for all the partitions considered, accuracy is lowest under the L_1-norm and increases until about $p = 3$.

[11]We considered alternative approaches to deal with this issue. Performing 20- or 30-cross validation would have been a reasonable choice only for the larger datasets, whereas leave-one-out was computationally impractical. Averaging over several partitions with 10-cross validation appeared to be a solution appropriate for all the datasets.

[12]The case $p = 1$, for which no previous solution is available, was started at the origin.

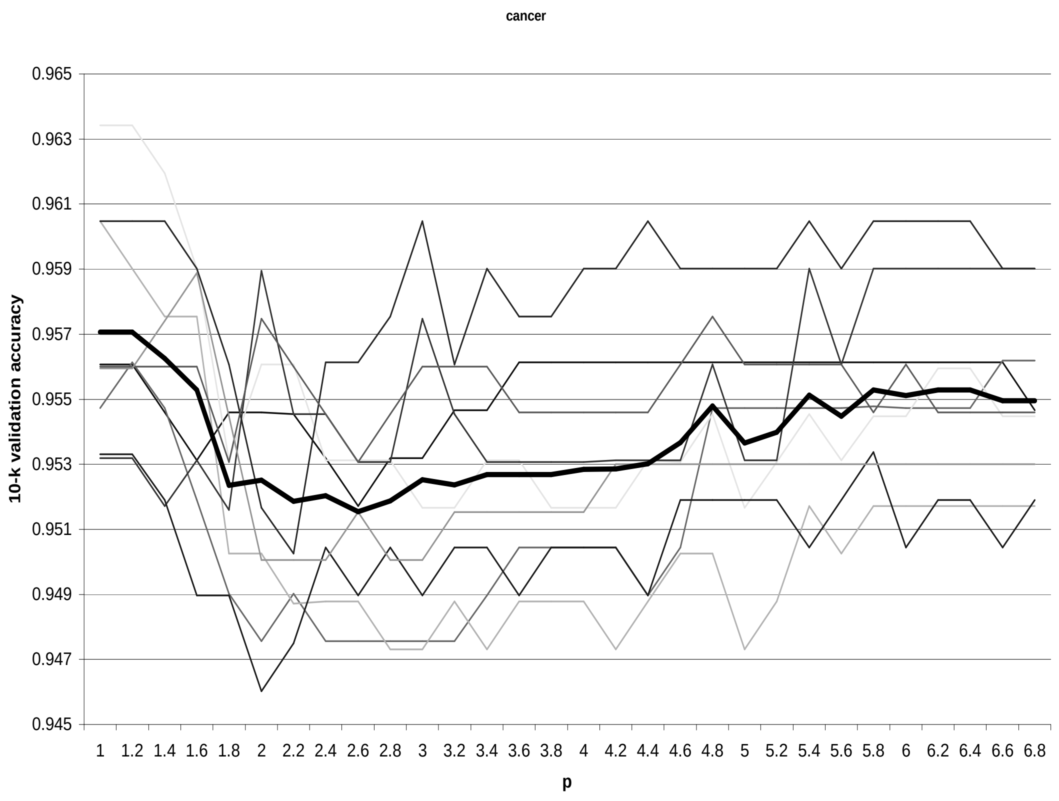

Figure 1.4: 10-fold validation accuracy vs. norm chosen: *Cancer* Dataset.

The choice of p appears thus to be relevant in some, but not all instances. A practical implication is that a range of values (including fractions) might be tried when L_p-norm separation is being considered.

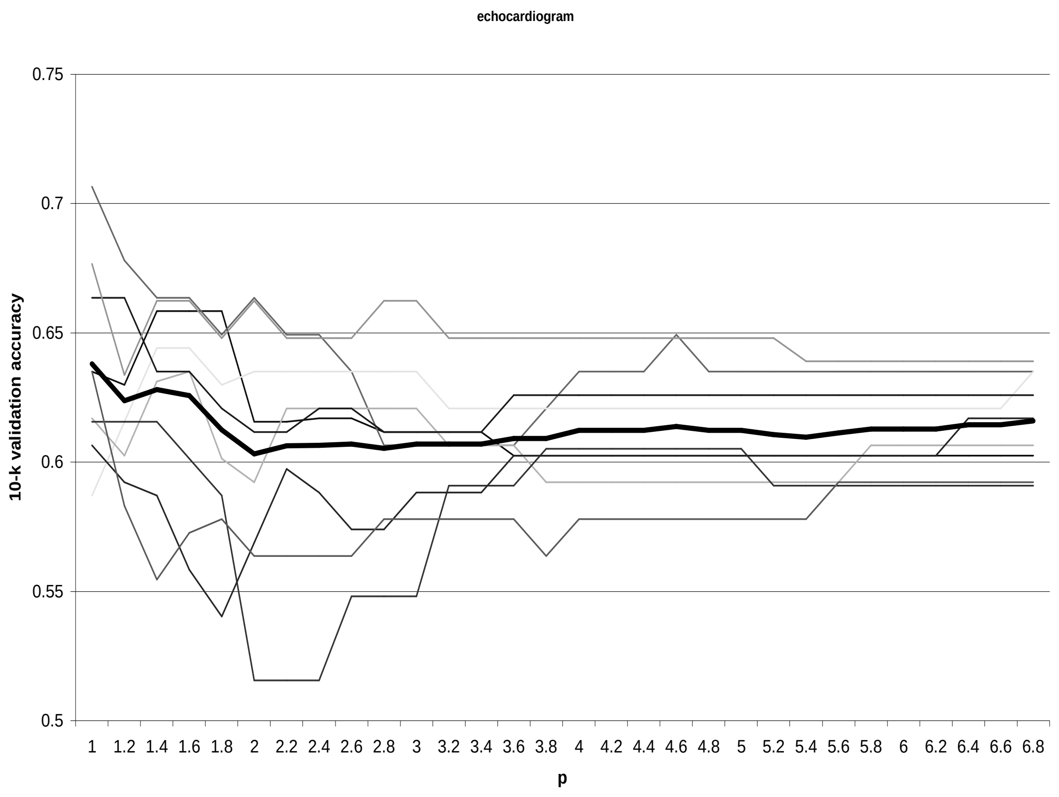

Figure 1.5: 10-fold validation accuracy vs. norm chosen: *Echocardiogram* Dataset.

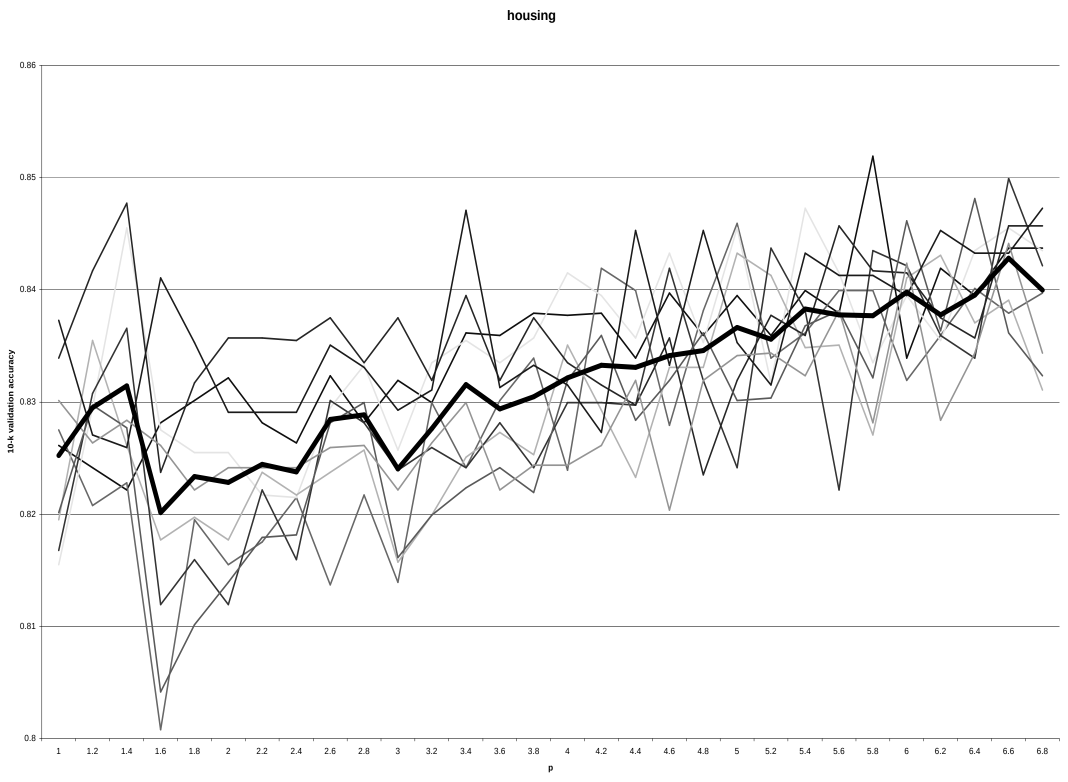

Figure 1.6: 10-fold validation accuracy vs. norm chosen: *Housing* Dataset.

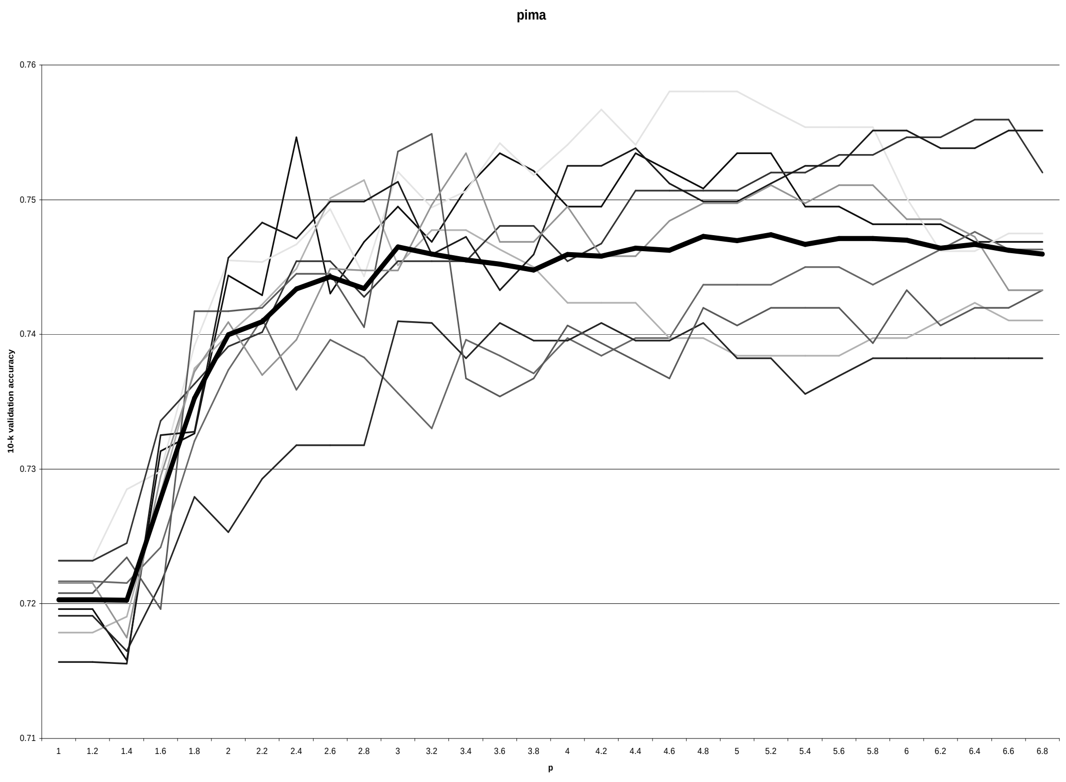

Figure 1.7: 10-fold validation accuracy vs. norm chosen: *Pima* Dataset.

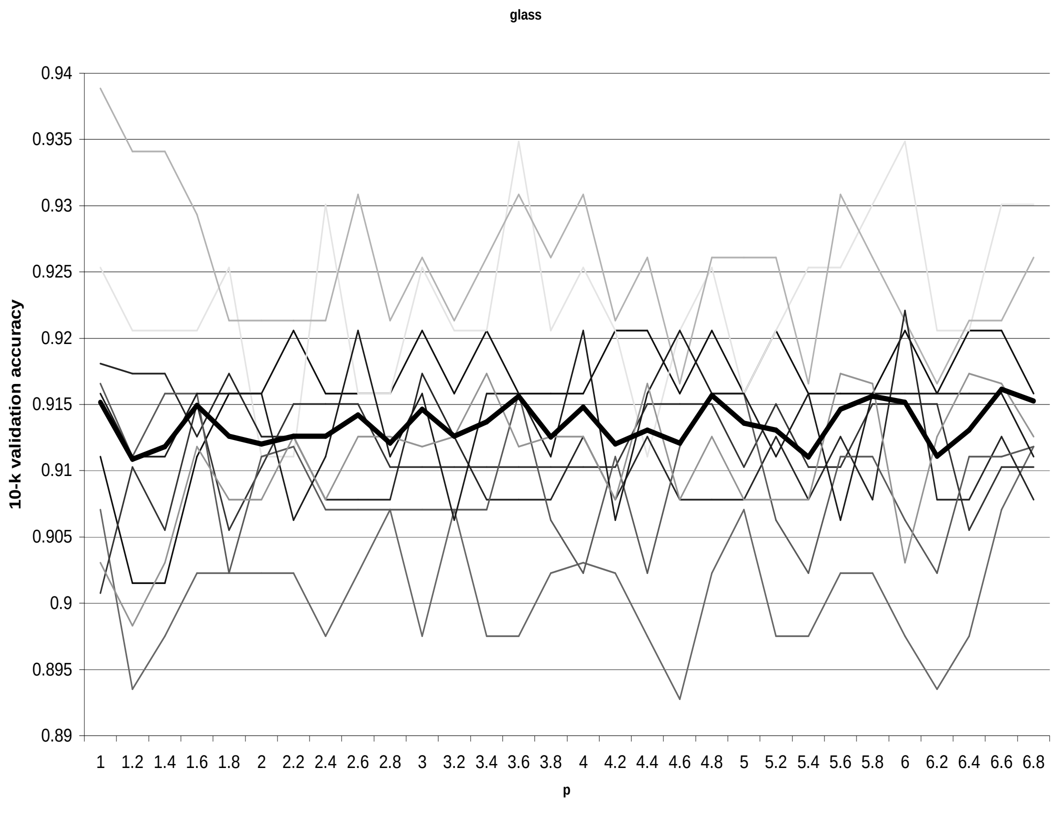

Figure 1.8: 10-fold validation accuracy vs. norm chosen: *Glass* Dataset.

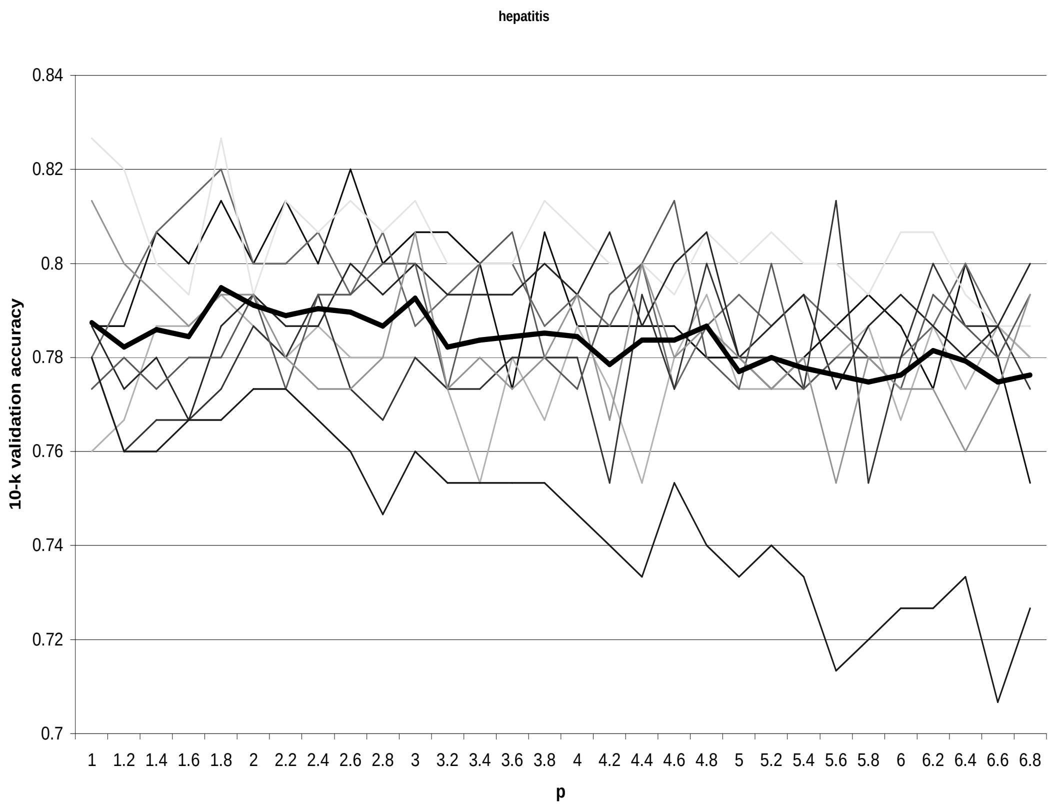

Figure 1.9: 10-fold validation accuracy vs. norm chosen: *Hepatitis* Dataset.

1.5 Conclusion

Our heuristic approach for the L_p-norm separating hyperplane problem appears to be very promising for data mining applications for which this criterion is chosen. It is able to find good solutions in reasonable time, for large instances in up to about 10 dimensions. The quality of the solutions found deteriorates in higher dimensions but appears to be robust to the number of observations. In moderate dimensions, the algorithm scales very gracefully (apparently linearly) to very large problems.

The proposed method can furthermore deal with problems for which no practical method for exact solution is known, namely when a truly arbitrary p is considered. Our exploration of the generalization properties of L_p-norm separation with respect to the choice of p suggest that there are classes of problems for which the norm matters, while for others the resulting hyperplane is essentially the same regardless of the norm.

Possible avenues for further work on the optimization phase include exploring the use of local search procedures other than the downhill simplex method within VNS, or of metaheuristic approaches other than VNS.

CHAPTER 2

EXACT SOLUTION OF L_∞-NORM AND L_2-NORM PLANE SEPARATION

2.1 Introduction

A basic problem in supervised classification is the separation of (or discrimination between) two sets of points in the real space $\mathbb{R}^n$ with a (hyper)plane that assigns a half-space to each of the sets. This problem is of interest in data mining and machine learning settings. When the interior of the convex hulls of the sets intersect, perfect linear separation is not possible, as there are no plane leaving all points from the first set on a side of the plane, and those of the other set on the opposite side. A discriminating plane may be found by minimizing some measure of the separation error, such as the total or average number of misclassified points, or some notion of distance of such points to the plane (for a historical overview, see [Sta97] and references therein). This often leads to difficult optimization challenges, and the separation criteria are sometimes chosen so as to keep the problem tractable.

The fairly intuitive objective of minimizing the sum of distances of misclassified points to the plane appears to have been behind several approaches to the separation problem, often leading to various Linear Programming formulations. The objective functions in these programs are, however, only substitutes for the true measures of the distances that ignore the essentially non-linear nature of the optimization problem.

In [Man99], Mangasarian states the problem precisely in terms of the analytical expression for the sum of L_p-norm distances of misclassified points to the plane. This results in a formulation of the general problem as that of minimizing a convex function (involving a sum of $\max\{0, \cdot\}$ operators) over a unit sphere in a norm dual to that one originally chosen to measure the distances from the misclassified points to the plane. An exact solution is often practical for $p = 1$, as for the L_1-norm the problem can be solved by $2n$ linear programs[1].

Our work takes these efforts a step further by exploring an exact solution ap-

[1] An alternative heuristic approach for this case is also presented in [Man99] which is a successive linearization algorithm applied to a penalty reformulation of the problem. It is indirectly suggested that this technique could also be used for the L_∞-norm case.

proach for the L_∞-norm, for which we propose and test a mixed integer formulation.

In order to probe the practical applicability, potential scalability and speed of our formulation, we test it on several publicly available data sets, as well as on four series of randomly generated problems. In the context of these tests, we also explore the effect of using heuristic bounds to accelerate our exact solution implementation.

The results for the L_∞-norm are compared to those corresponding to the exact solution to the L_2-norm problem, as obtained by Sylvain Perron [Per04] with an adaptation of the branch-and-cut algorithm of Audet *et al.* [AHJS00] for non-convex quadratically constrained quadratic programs, and reported in [AHK+04].

The rest of the chapter is organized as follows. The following section establishes the notation and summarizes some key results from [Man99]. Section 2.3 presents our formulation for the L_∞-norm. We then describe, in Section 2.4, the data sets used for our numerical tests and discuss the corresponding results. Section 2.5 concludes.

2.2 Notation and problem statement

We here establish some additional notation and state the general optimization problem.

For a fixed $p \in [1, \infty]$, the L_p-norm distance between a point $x \in \mathcal{A} \cup \mathcal{B}$ and its projection $\pi(x)$ to the plane P is given [Man99] by

$$\|x - \pi(x)\|_p = \frac{|w^t x - \gamma|}{\|w\|'_p} \tag{2.1}$$

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

$$\|\cdot\|'_2 = \|\cdot\|_2 \,, \|\cdot\|'_1 = \|\cdot\|_\infty \text{ and } \|\cdot\|'_\infty = \|\cdot\|_1 \,. \tag{2.2}$$

Although the formulae for the arbitrary-norm distance from a point to a hyperplane had been derived in other settings (e.g., [Mel97]), [Man99] appears to be the first to establish and use them explicitly in the context of linear discrimination.

Note that there is one degree of freedom in the characterization of the plane P, which can be used to fix an arbitrary scale. The choice of the scaling constraint

$$\|w\|_p' = 1 \tag{2.3}$$

removes the denominator from (2.1) and rules out the null solution $w = 0$ that has haunted some previous formulations of this problem. This approach, which appears to have been first proposed in [CIS89], is used in several studies (see, e.g., [MMS95] and references therein), and elegantly generalizes to the arbitrary-norm case.

The resulting optimization problem (problem(17) of [Man99]) is

$$\min_{w,\gamma} \left\{ \sum_{i=1}^{m} \max\left\{-w^t A_i + \gamma, 0\right\} + \sum_{j=1}^{k} \max\left\{w^t B_j - \gamma, 0\right\} \middle| \, \|w\|_p' = 1 \right\} \tag{2.4}$$

where $w \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}$.

This program can be reformulated in order to linearize the $\max\{\cdot, 0\}$ operators in the objective function. This results in

$$\min_{w,\gamma,y,z} \left\{ \sum_{i=1}^{m} y_i + \sum_{j=1}^{k} z_j \middle| \begin{array}{c} y_i \geq -w^t A_i + \gamma \text{ for } i = 1,..,m \\ z_j \geq w^t B_j - \gamma \text{ for } j = 1,..,k \\ \|w\|_p' = 1 \\ y \geq 0, \quad z \geq 0 \end{array} \right\} \tag{2.5}$$

where $w \in \mathbb{R}^n$, $\gamma \in \mathbb{R}$, $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^k$. The following sections are based on this last formulation.

2.3 L_∞-norm: A Mixed Integer Formulation

For the L_∞-norm case, the constraint (2.3) requires

$$\|w\|_\infty' = \|w\|_1 = \sum_{l=1}^{n} |w_l| = 1. \tag{2.6}$$

This can be addressed by the usual technique to linearize the absolute value operator. We replace w by two non-negative vectors w^+ and w^- in $\mathbb{R}^n_+$ such that

$$w^+ + w^- = |w| \tag{2.7}$$

$$w^+ - w^- = w \tag{2.8}$$

$$0 \leq w^+ \leq 1 \qquad 0 \leq w^- \leq 1. \tag{2.9}$$

We must also include a vector of binary variables $\delta \in \{0,1\}^n$ to force at most one of each pair of variables w_l^+ and w_l^- to be nonzero.

The resulting formulation is:

$$\min_{w^+,w^-,\gamma,\delta,y,z} \left\{ \sum_{i=1}^{m} y_i + \sum_{j=1}^{k} z_j \;\middle|\; \begin{array}{cc} y_i \geq (-w^+ + w^-)^t A_i + \gamma & \text{for } i = 1,..,m \\ z_j \geq (w^+ - w^-)^t B_j - \gamma & \text{for } j = 1,..,k \\ \sum_{l=1}^{n}(w_l^+ + w_l^-) = 1 & \\ w_l^+ \leq \delta_l & \text{for } l = 1,...,n \\ w_l^- \leq 1 - \delta_l & \text{for } l = 1,...,n \\ y \geq 0, \quad z \geq 0 & \\ w^+ \geq 0, \quad w^- \geq 0 & \\ \delta \in \{0,1\}^n & \end{array} \right\} \tag{2.10}$$

where $\gamma \in \mathbb{R}$, $y \in \mathbb{R}^m$, $z \in \mathbb{R}^k$, $w^+ \in \mathbb{R}^n$ and $w^- \in \mathbb{R}^n$.

This model is somehow similar to the one proposed in [Gle99], where a MIP approach is presented as an answer to the null solution problem[2]. The spirits of both approaches are clearly related. However, two significant differences should be pointed out. In the model presented in [Gle99]:

- Non zero w_i^- and w_i^+ variables are forced to be greater than a small but arbitrary parameter ϵ, and
- A classification gap is inadvertently introduced[3].

Under these circumstances, although the standardization constraints are similar, this model does not strictly address the problem of minimizing the sum of

[2]This reference came to our attention after having independently studied the proposed formulation. There is what appears to be a typographical error in [Gle99]; the inequalities in expressions (10e) and (10g) are meant to correspond to those in expressions (9b) and (9d), for which they would have to be inversed.

[3]This can be seen by comparing the signs of γ and d in expressions (10b) and (10c) of [Gle99] with the corresponding constraints in our formulation or in program (20) of [Man99]. We believe this to be accidental because the author later mentions that a gap *can* be forced.

L_∞-norm distances of misclassified points; it was developed in a different context and was never meant to do so. To the best of our knowledge, the generalization properties of this model have not been studied in the literature.

We now discuss the results of our empirical tests.

2.4 Numerical experiments

We tested our formulation (2.10) for the L_∞-norm case and our solution approach for program (2.5) for the L_2-norm case on a set of instances from the UCI Repository [DNM98] and two series of random problems created with D. Musicant's NDC generator [Mus98], which produces normally distributed clusters. This generator is publicly available and has been used in other discrimination studies (e.g., [FM01,MM00]). The parameters used for the generation of the artificial sets, as well as the preprocessing steps for the real life instances from the UCI Repository are detailed in the Appendix.

Heuristic approaches are being increasingly used as valuable tools to accelerate exact solution methods (see, e.g., [DDS02, HBUM03]). We measure the effect, on the performance of our implementations, of adding a heuristic bound. We first obtain heuristic solutions to all the problems with an application of Variable Neighborhood Search (VNS) [AHK+04], and use the objective value found as a cutoff limit in the branching processes.

The mixed integer programs for the L_∞-norm were solved directly with ILOG CPLEX 8.1, using default settings. All tests were performed on dual processor computers[4] running under Linux.

In all tables, the fit column is the full set classification accuracy of the solution found. The objective value is rounded to its approximate accuracy, namely 10^{-5}. Time is measured in CPU seconds, unless otherwise indicated. Time is also reported for the runs where the heuristic solution was used to accelerate the exact solution process. The last column shows the net time savings obtained by this procedure with respect to the solution time without the heuristic bound, considering of course the time used to obtain the heuristic solution[5]. It is interesting to point out that the VNS heuristic used found the exact solution, to within its accuracy, for most

[4]Intel Xeon 3.06 GHz, 1 Mb cache memory, 2 GO RAM.

[5]The savings are computed as $\frac{TWOB-TWB-TGB}{TWOB}$, where $TWOB$ is the exact solution time without heuristic bound, TWB is the solution time with the bound and TGB is the time used to get the bound (i.e., the VNS CPU time).

instances. For the others, the solution was quite close. The exact algorithms were thus performing mostly a confirmation exercise.

We first discuss the results obtained on the random problems, and then turn to the UCI real life instances.

2.4.1 Random problems

For the first series of random problems, we fixed the dimension at 6 and explored the effects of increasing the number of points from 2000 to 20000 (by steps of 2000). We then generated a second series of problems with 2000 observations, with dimensions ranging from 4 to 13. We will refer to these two test sets as NDC6d and NDC2k respectively. For each problem size, we generated 10 instances and report the corresponding mean values of the results in tables 2.1, 2.2, 2.3 and 2.4.

Acceleration by inclusion of a heuristic solution results in significant time savings on relatively large problems, for both the L_2-norm and the L_∞-norm cases. However, the inclusion of the heuristic solution is counterproductive in some cases.

Under the L_2-norm, this is due to the fact that the new bound is used to refine existing bounds (in nodes of depth up to five); the time spent on these refinements is not compensated for by the node reduction effect on problems with few dimensions. The net savings in higher dimensions, as well as in the UCI problems, are dramatic. In fact, instances in 12 and 13 dimensions could not be solved in reasonable time without the heuristic bound.

Under the L_∞-norm, inclusion of the heuristic bound results in net time gains in larger instances than for the L_2-norm case, because the exact solution time is relatively small with respect to the effort of obtaining the heuristic solution in the first place. Since exact solutions are found very quickly for all the UCI problems and the artificial problems with less than about 10000 points or about 10 dimensions, the heuristic enhancement is not appropriate. However, significant net savings are obtained for larger instances, and they increase with the size of the problem.

Table 2.1: L_∞-norm results on NDC2k series

problem size		VNS			MIP		MIP w/initial sol	
dim	obs	gap %	fit	CPU secs	fit	CPU secs	CPU secs	net time savings
4	2000	0.00%	94.29%	8.3	94.31%	0.8	0.3	-1363%
5	2000	0.00%	93.69%	7.1	93.71%	1.5	0.7	-366%
6	2000	0.00%	93.11%	6.9	93.14%	3.9	1.8	-166%
7	2000	0.00%	91.49%	9.4	91.54%	8.8	4.5	-80%
8	2000	0.00%	89.63%	14.0	89.69%	20.6	14.7	-49%
9	2000	0.05%	89.92%	21.4	90.06%	49.4	33.4	-10%
10	2000	0.00%	89.37%	28.2	89.46%	102.8	66.5	11%
11	2000	0.02%	83.48%	31.4	83.59%	330.9	281.1	5%
12	2000	0.04%	88.41%	42.3	88.58%	530.5	423.9	14%
13	2000	0.27%	85.56%	49.7	85.73%	1324.6	1213.9	8%

Table 2.2: L_∞-norm results on NDC6d series

problem size		VNS			MIP		MIP w/initial sol	
dim	obs	gap %	fit	CPU secs	fit	CPU secs	CPU secs	net time savings
6	2000	0.00%	93.10%	7.1	93.13%	3.7	1.7	-140%
6	4000	0.00%	91.90%	13.5	91.92%	14.9	7.8	-43%
6	6000	0.00%	91.51%	16.5	91.53%	40.9	23.3	3%
6	8000	0.00%	92.24%	33.8	92.25%	65.1	37.3	-9%
6	10000	0.00%	91.08%	45.2	91.08%	126.5	73.6	6%
6	12000	0.00%	90.94%	32.9	90.94%	204.0	114.0	28%
6	14000	0.00%	89.56%	37.2	89.58%	320.7	168.8	36%
6	16000	0.00%	92.21%	70.6	92.22%	288.1	160.4	20%
6	18000	0.00%	92.82%	52.1	92.82%	391.3	234.8	27%
6	20000	0.00%	94.49%	57.1	94.49%	368.0	202.8	29%

Differences between L_∞-norm and L_2-norm in full set accuracy do not appear to be very large. However, on both NDC series, the planes found with the L_2-norm criterion provided better fit than those obtained by minimizing the L_∞-norm.

Solution time grows exponentially with the dimension, while it appears to follow a power rule with respect to the number of observations. Best-fit details are provided in Appendix 2.6.

Under the L_2-norm, solution of instances of 20000 observations in 6 dimensions took an average of about 1.6 CPU hours, while the L_∞-norm problems on the same data sets where solved in an average of only about six CPU minutes.

Table 2.3: L_2-norm results on NDC2k series

problem size		VNS			EXACT		EXACT w/initial sol.	
dim	obs	gap %	fit	CPU secs	fit	CPU secs	CPU secs	net time savings
4	2000	0.00%	94.41%	10.5	94.41%	4.961	6.3	-238%
5	2000	0.00%	93.83%	11.1	93.84%	11.447	17.4	-149%
6	2000	0.00%	93.21%	13.9	93.23%	34.471	38.1	-51%
7	2000	0.00%	91.73%	21.8	91.75%	118.343	95.6	1%
8	2000	0.00%	90.13%	29.2	90.16%	557.403	223.1	55%
9	2000	0.00%	90.13%	44.3	90.17%	1796.287	680.1	60%
10	2000	0.00%	89.51%	56.8	89.55%	3194.37	1728.4	44%
11	2000	0.00%	83.65%	64	83.74%	38530.52	17491.7	54%
12	2000	0.01%	88.87%	78.5	89.02%	N/A	18970.8	N/A
13	2000	0.01%	86.02%	93.8	86.15%	N/A	60818.1	N/A

Table 2.4: L_2-norm results on NDC6d series

problem size		VNS			EXACT		EXACT w/initial sol.	
dim	obs	gap %	fit	CPU secs	fit	CPU secs	CPU secs	net time savings
6	2000	0.00%	93.14%	14.6	93.15%	36.0	38.2	-47%
6	4000	0.00%	92.22%	28.3	92.22%	212.6	210.6	-12%
6	6000	0.00%	91.59%	34.2	91.60%	635.5	726.1	-20%
6	8000	0.00%	92.27%	54.9	92.28%	959.7	947.1	-4%
6	10000	0.00%	91.04%	77.7	91.04%	1433.3	1950.2	-41%
6	12000	0.00%	91.01%	69.6	91.01%	2593.5	2131.5	15%
6	14000	0.00%	89.68%	74.9	89.69%	2858.2	3877.6	-38%
6	16000	0.00%	92.25%	112	92.25%	2146.0	3143.3	-52%
6	18000	0.00%	92.80%	116	92.80%	5915.7	5700.6	2%
6	20000	0.00%	94.59%	119	94.59%	5777.2	3508.1	37%

As exact solution of L_∞-norm problems is considerably faster than that of L_2-norm problems, we considered worthwhile to test the behavior of our L_∞-norm method on larger problems, for which averaging over several instances would be impractical. We therefore generated two additional series of unique problems, with simpler structure and for which the complexity (as approximated by the number of full-set missclassifications) could be easily controlled. The first set of these additional test problems, denoted Sym2k , considers instances with 2000 observations in 14, 16 and 18 dimensions, while the set which we call Sym6d includes instances in 6 dimensions with up to 100000 points by increments of 10000.

For the simpler, more precisely controlled instances of the Sym2k and Sym6d series (used only with the L_∞-norm criterion), the objective function grows monotonically, as expected, with the problem size. Note, however, that the average objective function values for the NDC series showed some fluctuations, reflecting the greater variability of those problems.

Table 2.5: L_∞-norm results on Sym2k series

problem size		VNS			MIP		MIP w/initial sol	
dim	obs	gap %	fit	CPU secs	fit	CPU hrs	CPU hrs	net time savings
14	2000	0.00%	92.00%	34.1	92.20%	0.6	0.5	12%
16	2000	0.01%	91.90%	56.7	92.10%	2.7	2.5	4%
18	2000	0.00%	89.95%	86.5	90.20%	16.9	13.8	18%
20	2000	0.02%	91.95%	106.8	92.15%	81.5	60.2	26%

Table 2.6: L_∞-norm results on Sym6k series

problem size		VNS			MIP		MIP w/initial sol	
dim	obs	gap %	fit	CPU secs	fit	CPU secs	CPU secs	net time savings
6	10000	0.00%	88.95%	17.1	88.96%	156.3	96.7	27%
6	20000	0.00%	88.73%	36.4	88.74%	786.4	461.4	37%
6	30000	0.00%	89.10%	50.1	89.11%	1774.4	1197.7	30%
6	40000	0.00%	89.01%	78.8	89.01%	3972.7	2326.4	39%
6	50000	0.00%	88.95%	89.8	88.95%	6004.2	3428.2	41%
6	60000	0.00%	89.06%	133.6	89.07%	7655.7	5050.1	32%
6	70000	0.00%	89.22%	126.6	89.23%	15215.0	7495.1	50%
6	80000	0.00%	88.86%	169.2	88.86%	21864.7	10639.1	51%
6	90000	0.00%	89.02%	175.9	89.02%	20072.5	11650.6	41%
6	100000	0.00%	88.98%	177.1	88.99%	35206.3	14020.3	60%

In addition to the four random problem series discussed above, we solved, for the L_∞-norm, a single instance of 100000 points in 10 dimensions, generated with the same criteria as the Sym6d and Sym2k problem series. The solution time, accelerated by the inclusion of the heuristic solution, was about 84 CPU hours.

2.4.2 Real life instances

Tables 2.7 and 2.8 present the results obtained for the UCI problems, under the L_∞-norm and the L_2-norm, respectively. These instances are considerably smaller than our random problems, and the use of a heuristic bound does not yield net time savings in the exact solution under the L_∞-norm. Under the L_2-norm, however, the time savings are very significant in all but one problem.

Table 2.7: L_∞-norm results on UCI instances

problem	problem size		VNS			MIP		MIP w/initial sol	
	dim	obs	gap %	fit	CPU secs	fit	CPU secs	CPU secs	net time savings
cancer	9	683	0.01%	97.37%	7.6	97.51%	1.1	1.1	-694%
diabetes	8	768	0.00%	76.04%	4.4	76.04%	8.1	7.7	-50%
echocardiogram	6	74	0.00%	72.97%	0.5	77.42%	0.1	0.1	-440%
glass windows	9	214	0.82%	94.39%	2.8	95.79%	0.4	0.4	-657%
hepatitis	16	150	3.49%	89.33%	7.4	90.00%	65.7	63.5	-8%
housing	13	506	4.50%	86.76%	13.2	88.14%	30.4	29.1	-39%

Table 2.8: L_2-norm results on UCI instances

problem	problem size		VNS			EXACT		EXACT w/initial sol.	
	dim	obs	gap %	fit	CPU secs	fit	CPU secs	CPU secs	net time savings
cancer	9	683	0.50%	97.07%	6.2	97.07%	156.2	57.0	60%
pima	8	768	0.00%	75.39%	4.4	75.39%	611.6	211.2	65%
echocardiogram	6	74	0.03%	75.68%	0.6	75.68%	4.6	1.3	58%
glass	9	214	0.79%	94.86%	3.5	95.33%	4.5	1.4	-8%
housing	13	506	0.57%	84.19%	13.9	84.39%	550.2	30.4	92%
hepatitis	16	150	1.17%	86.67%	4.4	88.00%	14181.3	50.7	99.6%

2.4.3 Cross validation

The techniques proposed in this paper allow us to compare the classification performance of separating planes obtained by minimizing different norms. Figure 2.1 summarizes the full-set fit for the UCI problems under the L_2- and L_∞-norms. We also include, for comparison, the results under the L_1-norm obtained by solving the corresponding $2n$ linear programs (see [Man99]). Figure 2.2 shows the 10-cross validation mean accuracy of the planes found under the three norms.

To our surprise, the plane obtained under the L_∞-norm performs at least as well as the others, both in full set and 10-cross accuracy, for all problems except *echocardiogram*, which is by far the smallest dataset we consider, with only 74 points.

We note that the L_1-norm plane, which is considerably easier to compute than the others, performs better than the L_2-norm plane in three cases and nearly as well in two others. This suggests that the L_1-norm could be a good bet on large problems for which the alternatives are impractical.

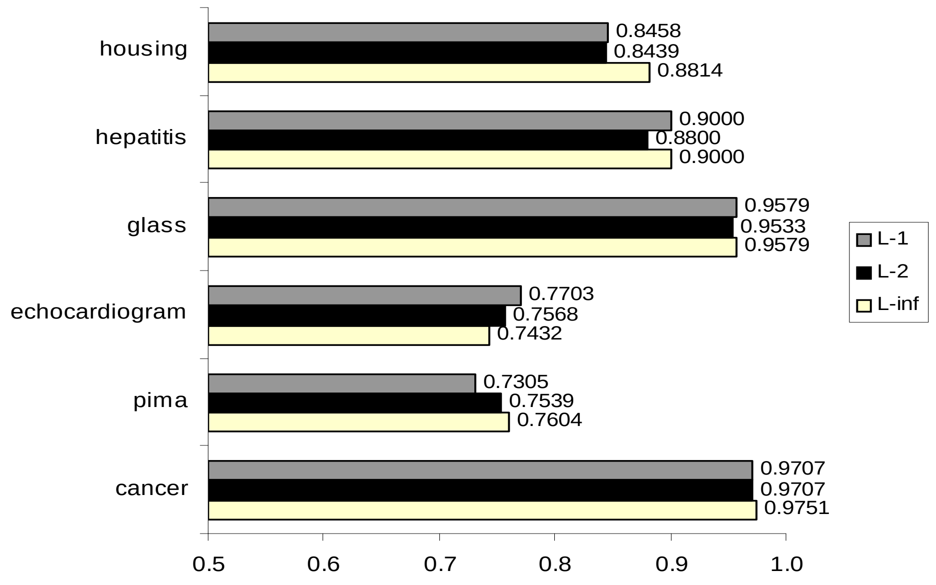

Figure 2.1: Full set accuracy

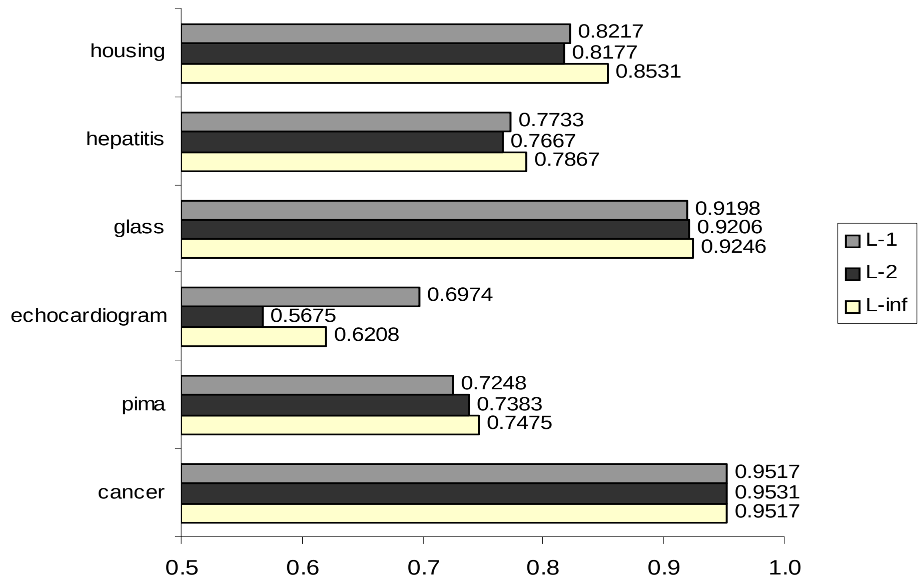

Figure 2.2: 10-cross validation

2.5 Conclusion

L_p-norm separation, a classic problem in supervised classification, presents important optimization challenges. Despite recent progress, practical techniques for the exact solution of cases other than the L_1-norm have remained unavailable.

We propose and implement an approach for the exact solution of fairly large problems in L_∞-norm. We solve in reasonable computing times examples of up to 20000 points (in 6 dimensions) and 13 dimensions (with 2000 points). A solution was also found for an example of 100000 points in 10 dimensions.

We also show that, for sufficiently large problems, computation times for the L_2 and L_∞-norms can be substantially reduced by incorporating heuristic results in the exact solution process.

Several real-life instances from the UCI Repository are also considered, and we are able to compute and compare full set fit and generalization properties (as estimated by 10-cross validation) for the L_1, L_2 and L_∞-norms.

2.6 Appendix: Details on Curve Fitting for Solution Time Behavior

Among the models tried, the best fit for the case with 2000 points was obtained by $t = \kappa_1 \cdot e^{\kappa_2 \cdot n}$, where t is CPU time in seconds. For the problems on 6 dimensions, the best curve was $t = \kappa_3 \cdot (m+k)^{\kappa_4}$, where $m+k$ is the total number of points. The estimation details are as follows. For the L_∞-norm:

- $\kappa_1 = .0336$, $\kappa_2 = .8004$, $R^2 = .998$
- $\kappa_3 = .00000042$, $\kappa_4 = 2.3683$, $R^2 = .994$

For the L_2-norm:

- $\kappa_1 = .0246$, $\kappa_2 = 1.2409$, $R^2 = .985$
- $\kappa_3 = .0000044$, $\kappa_4 = 2.1267$, $R^2 = .974$

These estimates where performed considering together the data from NDC and Sym tables, without heuristic acceleration.

CHAPTER 3

MISCLASSIFICATION MINIMIZATION: EXACT AND HEURISTIC APPROACHES

3.1 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]).

3.2 Formulation

3.2.1 Statement of the Problem

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

$$\min_{w,\gamma,y,z} \left\{ \sum_{i=1}^{m} y_i + \sum_{j=1}^{k} z_j \;\middle|\; \begin{array}{c} -w^t A_i + \gamma \leq M y_i \text{ for } i = 1, .., m \\ w^t B_j - \gamma \leq M z_j \text{ for } j = 1, .., k \\ y \in \{0,1\}^m, z \in \{0,1\}^k, w \in \mathbb{R}^n, \gamma \in \mathbb{R} \end{array} \right\} \tag{3.1}$$

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 here with these variations and, for comparability, will only consider the basic objective of minimizing the total number of misclassified points.

3.2.2 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 $[\gamma \quad w] = 0$, 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

[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 on 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 chapter 5.

[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.

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.

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 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 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

[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 $\|\cdot\|'_\infty = \|\cdot\|_1$.

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

[7]See chapter 1, 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 3.2.1, we ignore these weights.

the constraints

$$\begin{aligned} -1 + 2D_l \leq w_l \leq 1 - 2E_l \qquad l = 1, \ldots, n \qquad (3.2) \\ \sum_{l=1}^{n} D_l + \sum_{l=1}^{n} E_l = 1 \qquad (3.3) \\ E \in \{0,1\}^n \quad D \in \{0,1\}^n. \end{aligned}$$

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

$$M = 2n \max \left(\max_{\substack{i=1,\ldots,m \\ l=1,\ldots,n}} |A_{il}| \quad , \quad \max_{\substack{j=1,\ldots,k \\ l=1,\ldots,n}} |B_{jl}| \right). \qquad (3.4)$$

We will assume that the data in A and B have been rescaled to the range $[-1, 1]$. Linear rescaling is a quite standard procedure in practice, and all of our databases conform to it[9].

3.2.3 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

$$\begin{aligned} M_i^A = n + n \max_{l=1,\ldots,n} |A_{il}| \qquad i = 1, \ldots, m \qquad (3.5) \\ M_j^B = n + n \max_{l=1,\ldots,n} |B_{jl}| \qquad j = 1, \ldots, k \end{aligned}$$

[9]Note that the data used in the other chapters is rescaled to $[0, 1]$. The choice of a different range (i.e., $[-1, 1]$), for the experiments reported in this chapter was made because we suspected that this larger range might improve numeric performance, while it obviously respects the assumptions of the model. We did not actually test the hypothesis that performance is indeed better under the alternative rescaling.

in the corresponding constraints, instead of the common M. The validity of these bounds can be seen by an argument similar to that presented in [LW78]. Its adaptation to our model can be intuitively summarized as follows. Since $-1 \leq w_l \leq 1$, it follows that for any $i = 1, \ldots, m$,

$$w^t A_i \leq n \max_{l=1,\ldots,n} |A_{il}| \tag{3.6}$$

$$w^t B_i \leq n \max_{l=1,\ldots,n} |B_{il}| . \tag{3.7}$$

Also note that

$$\gamma \leq \max \left(\max_{i=1,\ldots,m} w^t A_i \quad , \max_{j=1,\ldots,k} w^t B_j \right) \leq n$$

where the second inequality follows from the fact that the matrices A and B have been rescaled to the range $[-1, 1]$. We therefore have that the first n in the right hand sides of 3.5 covers the worst case for γ and the terms $n \max_{l=1,\ldots,n} |A_{il}|$ and $n \max_{l=1,\ldots,n} |B_{il}|$ take care of the highest possible absolute values of $w^t A_i$ and $w^t B_j$, respectively.

Our final formulation is then

$$\min_{w,\gamma,y,z} \left\{ \sum_{i=1}^{m} y_i + \sum_{j=1}^{k} z_j \;\middle|\; \begin{array}{c} -w^t A_i + \gamma \leq M_i^A y_i \text{ for } i = 1, .., m \\ w^t B_j - \gamma \leq M_i^B z_j \text{ for } j = 1, .., k \\ -1 + 2D_l \leq w_l \leq 1 - 2E_l \\ \sum_{l=1}^{n} D_l + \sum_{l=1}^{n} E_l = 1 \end{array} \right\} \tag{3.8}$$

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 (3.5).

3.3 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, 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.

The space of directions is searched with a heuristic framework known as Variable Neighborhood Search (VNS) [HM01b], which has been applied successfully in a number of global optimization tasks (e.g., [BM96, CH00, HM97, BCK99, HMU04, MJPČ03]).

The algorithm for the oracle differs from that described in section 1.2 in that, for a given direction, it returns the minimum possible number of misclassified points, instead of the sum of L_p-norm distances. It is outlined in Figure 3.1.

The pseudo-code for the VNS search is the same as in figure 1.2, with the function PNORMDIST($\widehat{\alpha}$) replaced by COUNTMISSED($\widehat{\alpha}$).

```
• compute $distA$ and $distB$ /* vectors of projections to ray $\alpha$ */
• 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 3.1: Pseudocode for the COUNTMISSED(α) oracle.

We now present the results of our numerical experiments.

3.4 Numerical Experiments

We try our exact formulation and our heuristic on a small set of problems from the famous UCI Machine Learning repository [DNM98], 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 5.8.

Our exact solutions were obtained using the CPLEX callable library [ILO03], running under Linux[10]. Some of the problems we originally considered, such as the *Pima* database, could not be solved within the memory constraints of our solution setup[11].

We first compare the solution time for the LW model with our improved formulation, model (3.8) above, refered to as LWtight in the tables). 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.

3.4.1 Exact Solutions

Table 3.1 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[12]. 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 3.2 shows the corresponding results for the five random problems with 300 observations in 6 dimensions. In this case as well, the savings seem quite significant.

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

[11]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.

[12]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 3.1: CPU times for LW and LWtight formulations: UCI problems

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 3.2: CPU times for LW and LWtight formulations: NDC problems

3.4.2 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 3.3 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

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 3.3: Full set accuracy and VNS running times: NDC problems

takes the exact algorithm to reach an accuracy of about 98%. Whether the trade off is worthwhile will depend 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[13].

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[14]. In order to have a reference to another, mathematical programing 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 3.2. 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[15]. 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

[13]Furthermore, the full set fit for *Hepatitis* is also quite poor in the heuristic solution.

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

[15]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.

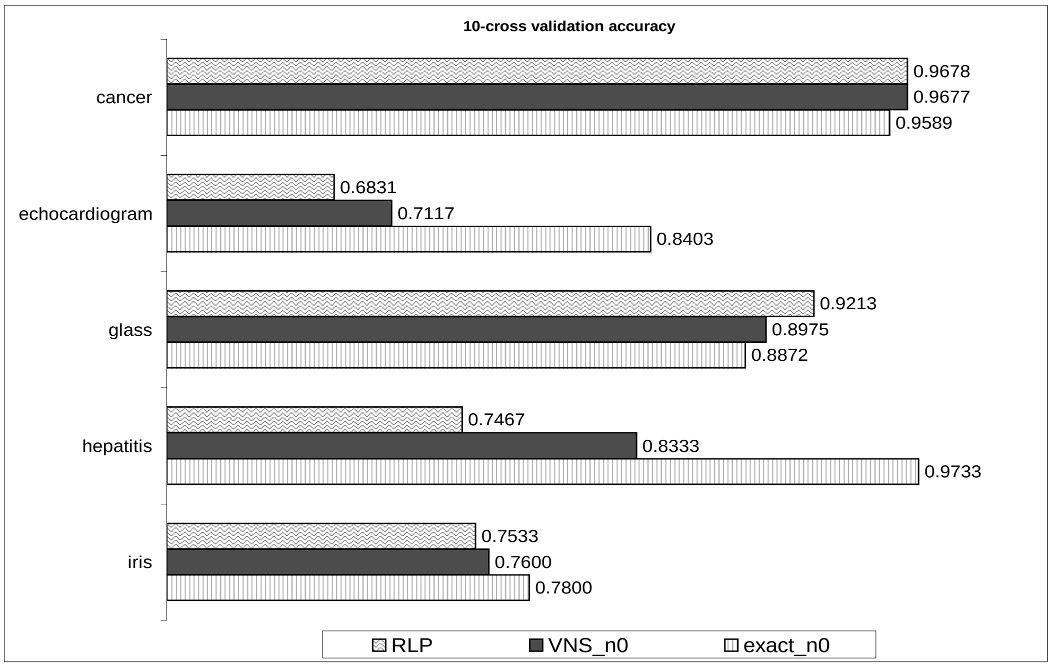

Figure 3.2: Generalization accuracy for alternative linear discriminants

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 3.3 shows the average running time of the ten random problems of each size, for a constant stopping criterion. On this range of 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 this kind of problems 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.

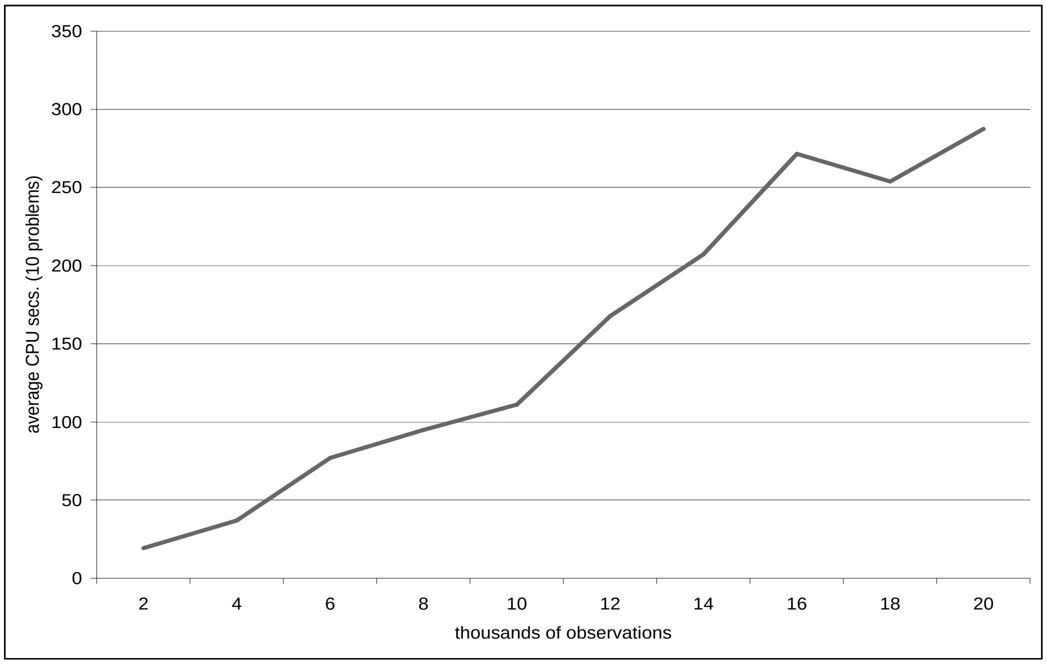

Figure 3.3: VNS solution time for NDC random problems in 6 dimensions.

3.5 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.

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 promising and to provide a base for further research. In particular it would be interesting to try the proposed exact formulation along with some of the algorithmic improvements introduced in the literature for this family of Mixed Integer Programs. The heuristic can be also benchmarked to alternative heuristic approaches for the classification minimization problem and tried on other sets of databases.

CHAPTER 4

ISSUES IN CONSUMER CREDIT SCORING

4.1 Introduction

In order to set up some thematic background on research about credit scoring, and provide a broader context for the following essay, this chapter presents a very brief review of issues, ideas, and references on the application of supervised classification models to the prediction of consumer credit behavior.

Since the seminal work of Altman in the late 1960's [Alt68] the applications of various discrimination techniques to corporate bankruptcy has been described in an overwhelming body of literature. Consumer default modeling, however, was mostly ignored in scholarly publications until very recently. Academic interest in the field has dramatically increased in recent years and a large number of journal articles and working papers are now available.

This phenomenon can probably be traced to several causes:

- The consumer lending industry itself has exploded in the last 20 years, and the demand for scoring technologies and related services has been pulled by it. As an example, Fair Isaac, the leader in the market for credit scoring systems, had gross revenues of only US$329 million in 2001, but made nearly US$600 million in 2004.

- As with other Data Mining applications, availability of huge databases and increased computing power opens new analytical opportunities and challenges.

- Unlike the financial ratios used to forecast corporate bankruptcy, consumer credit records are not publicly available. This is perhaps the single most important limitation to academic work on this subject. Increasingly, empirical work with proprietary data is reported, with some limitations, in scholarly journals. This is now the standard practice in the field, and one may wonder if it would have been considered acceptable 20 years ago.

Along the credit scoring process, one finds most of the usual challenges in any data mining task: data cleaning and preprocessing, feature selection, model selection and performance assessment. However, some issues (such as reject inference), have greater relevance than in other applications and others (e.g., population drift) display structural characteristics particular to the field.

Several surveys provide overviews of the issues involved in credit scoring, from various perspectives (ex. [RCW83,RG94,HH97,Han01,Tho00]). Hand's book [Han97], provides a summarizing overview of practical classification, and includes a section discussing credit scoring. Some technical papers prepared in the context of doctoral dissertations also survey the field ([Liu02]).

The following sections enumerate briefly some of the key issues in the construction of a consumer credit classification system.

4.2 Endogenous class definitions

A peculiar feature of the credit scoring problem is that the definition of the classes is itself a relevant part of the modeling effort. The observations on which models are built are usually existing portfolios of loans, some of which are defined as "good" or "performing" and others as "bad" or "non-performing". Although class definitions are often exogenously imposed by the circumstances (legal, regulatory, traditions, etc.) to the classification model, they are by no means obvious from the structural nature of the problem. A very common definition of a non-performing loan is one that has missed 3 consecutive payments. However, other reasonable characterizations can be defended. For example, [DAG96] used the definition of four consecutive missed payments, because it was the one adopted by the Spanish bank that provided their database.

This situation seems to be in contrast with some other examples of applications where the classes themselves are somehow natural or not subject to much ambiguity (is the tumor cancerous or not, is the character an "A" or a "B", is the wine from region X or Y, etc.). Of course this "structural" fuzziness in class definitions is not exclusive of this problem. For example, among the problems in the Irvine Repository, at least the Boston Housing database (where the threshold value for class definition of $21,000 seems quite arbitrary) is a similar case.

In the context of corporate bankruptcy, the issue of class definitions has received considerable attention (e.x. [Hay03] and numerous references therein), but remains poorly explored for consumer models.

4.3 Sample bias and reject inference

The “labeled” loans used to train the typical credit scoring system entered the institution’s portfolio because they where selected from a pool of applications by another credit scoring system or by any other selection mechanism. They are not, therefore, representative of the applicant population. The labels of the applications that where not accepted are usually unknown. Some statistical techniques are used to try to compensate for the resulting bias. Usually, some assumption about the prior probabilities of the classes or about the distribution of the characteristics is required. This is known in the literature as reject inference.

The sample bias phenomenon and the reject inference techniques proposed to address it have been extensively discussed in the credit scoring literature ([SG89, Joa94, AC01, BCT03, CB04, VVDP04]). The theoretical and empirical advantages of using reject inference techniques is at best doubtful, but they are often included in best practice implementations.

4.4 Mixed variable types

Credit scoring databases invariably contain both numeric (age, income, etc.) and categorical variables (gender, product category, location, etc.), so the usual difficulties of dealing with mixed variable types arise.

Some distance-based methods could theoretically incorporate numeric and categorical variables in an appropriate definition of the distance, as is done in some clustering applications (e.g., [Hua98]). In practice, however, the usual approach is to convert variables of one kind into the other. Some techniques to assign numerical weights to categorical variables are discussed in [Han97] and applied, for instance, in [HH96]. In credit scoring applications, the most common choice appears to be categorizing numeric variables, and several alternative algorithms for this purpose are discussed in the literature.

The most straightforward ways of partitioning the range of a continuous variable in order to recode it as a categorical one are either to take equal intervals or to consider the cut points that correspond to the percentiles of the desired number of intervals. As these methods do not take into consideration the distribution of the variable conditional to the class to which an observation belongs, their performance is bound to be poor. They are, however, sometimes used in practice (and, in the pattern recognition jargon, belong to the “unsupervised categorization” family of algorithms).

Numerous, more sophisticated, approaches have been proposed for this task. A famous one, based on information theory, is due to Fayyad and Irani ([FI92]), and is implemented in several software packages. A linear programming approach is proposed in [Gle04], while a simulated annealing algorithm is presented in [HA00].

A heuristic commonly used in the industry is to start with a fine (unsupervised) partition and then manually join a sufficient number of adjacent categories in order to smooth the weights of evidence of the resulting categories.

4.5 Model selection

As in most other classification domains, an important section of the literature is devoted to proposing and comparing alternative models. Of course, most researchers find that their method outperforms the others.

Several serious studies tend to find that differences between good implementations of different methods are often not dramatically large, and that the choice of the best model is database dependent. The structure of credit scoring problems does not therefore appear to be systematically suited for a particular family of prediction models (e.g., [BVGV$^+$03] and references therein).

In practice, regression-based models (logit, probit, etc.) are perhaps the most commonly implemented, along with linear programming based models.

4.6 Feature selection

A fundamental problem in any data mining application, feature selection is of course central to the credit scoring process. An initial set of 200-700 variables is common. However, in order to capture interactions among some variables (reflecting non-linearities in the decision frontier), they must be combined to generate additional dimensions. The resulting variables are sometimes called "derogatory trees" ([Han97]). The challenge is thus to select the few best among at least several hundred (often thousands) of alternatives.

The most widely used methods belong to the stepwise family. Variables are inserted and removed, usually one at a time, according to various criteria, until some stopping rule is met. These are, of course, local descent heuristics which work around the combinatorial challenge of approximating a global solution.

Some more or less sophisticated heuristic implementations have been proposed to explore the attribute space (e.g., [DMS01, BLSvW96]), but are only tried on very small instances and appear to be of limited applicability in practice.

Some recently proposed variations of the stepwise approach seem to be promising ([Pir99, Pir04] and specially [FS04]), specially in the context of statistical methods. There is, however, an important link between model choice and variable selection. Most of the literature on feature selection for credit scoring focuses on trees, logistic or regression models. The 1988 paper by Nath & Jones [NJ88] stands out as one of the very few, and perhaps the earliest discussions specifically oriented to LP based scoring models; it is discussed at length in chapter 5.

4.7 Population drift

Unlike many problems in the natural sciences, the structure in the credit classification problem is believed to change over time, as new credit applicants come from a population that is not appropriately represented by the sample used to build the model. This makes the choice of the observations used to train the system an important issue. On the one hand, one would like loans of different ages and maturities represented. In particular, it is important to have "seasoned" as well as relatively new loans in the training sample. However, the variables that best predict the performance of old loans may not be the same for the population from which new applications are drawn.

This issue is discussed, for example in [Tho00, HH97], and [HH96] point out that an advantage of nearest-neighbor approaches is that the training set can be smoothly updated by adding new cases and removing the oldest ones.

4.8 Application vs. behavioral scores

In the industry jargon, credit scorecards are often distinguished as to whether they are used for the original decision to grant a loan (application scoring) or to assess the ongoing status of an existing client (behavioral scoring). The latter can be used to predict credit quality (just like the application scores, but including internal data on the performance of the client), or to determine a level of collection action (such as letter, call or referral to legal team) or marketing action (cross selling of products, etc.).

The crucial difference from the analytical point of view is the panel-like structure of the data. Examples of recent references on this issue include [TH03], [AHT01] and [KS04].

CHAPTER 5

LINEAR PROGRAMMING APPROACHES TO CREDIT SCORING

5.1 Introduction

We consider the problem of discriminating credit applicants into two classes with different expected repayment behaviors. Although we base our discussion and our case study on the ranking of credit applicants, identical models are applied to other binary classification tasks in credit risk management, marketing, consumer relationship management, etc. Typical examples include rating loans in an existing portfolio, targeting offers or promotions to groups of consumers most likely to react to them, and detecting clients most likely to churn.

Many pattern classification techniques have been applied to this problem [RCW83, DEG92, RG94, HH97, Tho00, Han01]. It is sometimes desirable to produce, for each customer, a *score*, which is a scalar monotonically related to its predicted probability of belonging to a certain class. Furthermore, for many business processes, one would like to be able to compute this score as a weighted sum of observable variables (or at most a simple combination of them). The weight vectors in question are known as *scorecards*, and they characterize the family of linear discriminants.

Although some implementations of techniques such as neural networks or classification trees can be adapted to produce scores, in practice the most popular families of models for estimating them directly are regression (linear, logit, probit, etc.) and Linear Programming (LP). Our work focuses on the latter.

Since Fisher's seminal paper [Fis36] appeared in 1936, linear discriminants of various flavors have been developed by statisticians and applied in numerous fields. The introduction of discrimination by linear programing to the Operations Research literature can perhaps be credited to [Man65], although the fundamental ideas can be traced a few years earlier [Min61, Cha64]. The subject did not receive much academic attention for over a decade, but a paper published in 1981 by Freed and Glover's [FG81b] triggered a rich vein of literature on the subject. LP based discrimination is nowadays recognized as a powerful alternative in a data miner's toolbox, and is the engine behind some successful scoring systems implemented in the industry.

We study the optimal choice of variables to be considered in constructing the discriminant, from the set of available characteristics (a problem known as feature selection), within the context of discrimination by LP. We will refer indistinctly to the variables available for each observation, once properly codified, as characteristics, features or columns[1]. Feature selection (FS) is one of the key challenges in pattern recognition and an enormous literature on the subject exists. An appropriate choice of a subset of the available features has several advantages (limiting computational burden, easing interpretability, etc.), but the most important in many applications, and the one we will focus on, is the effect of variable choice on the generalization properties of the classifier. Too many features will overfit the training data and perform poorly on new observations; on the other hand, too few features will fail to capture the structure of the data and will not generalize well either[2].

In all problems of practical interest, with more than a few initial features to consider, the combinatorial challenge of choosing the best (in the sense of truly optimal) subset of features for a given classification task is overwhelming, and most research on this field develops and compares various heuristics[3].

Some FS algorithms consider correlations amongst the features and between the features and the training set class labels, without reference to a particular prediction model. These methods are often referred to as *filters*. On the other hand, there are FS algorithms, known as *wrappers*, which assess the performance of a specific classifier with a given subset of features, and then search the combinatorial space of columns for a (local) optimum[4].

Various FS methods are used in practical credit scoring applications. The filters commonly used include the χ^2 and the *information statistic* [TEC02], while the wrappers of choice belong to the family of stepwise local-search heuristics where a single (or a small number) of features is added and/or removed at each step. Stepwise algorithms (including backward, forward or hybrid) are included in many software packages and are used extremely frequently to estimate regression models[5].

In principle, any filter method can be used to select a subset of columns to use

[1]They would be called independent variables in statistics.

[2]For a full discussion of this issue, in the context of an application to a credit database, see [GT00].

[3]The difficulty of the combinatorial task, even in moderately sized problems, is perhaps better understood by noting that implicit enumeration approaches are not usually possible due to the lack of monotonicity in the most reasonable performance measures, i.e., the most useful set k columns does not necessarily contain the best set of $k-1$ columns and so on [Tou71].

[4]The now common denomination of filters and wrappers was apparently introduced by Kohavi and John [JKP94, KJ97].

[5]Sophisticated variants have been developed and applied to credit scoring applications (see e.g., [FS04]).

in an LP based classification model, and many wrapper algorithms can be applied using an LP model as the target performance method. The specific properties of LP models with respect to feature selection have, however, received relatively little attention in the literature. Nath and Jones [NJ88] were perhaps the first to address the issue of feature selection specifically in the context of LP models. We discuss, reinterpret and adapt their idea, and explore its performance on two real-life credit databases. We find that it outperforms other approaches, including common filter methods.

The use of proprietary databases in academic research on credit scoring has become increasingly common in recent years[6]. The impossibility of disclosing many details of the data and of any independent replication are compensated by the potential practical relevance of the results, which is naturally more questionable in work with the extremely limited and poor databases on consumer credit that are publicly available.

The rest of the paper is structured as follows. The next section presents the LP formulation that we implement for our study, placing it in the context of the family of models to which it belongs. Section 5.3 discusses the assessment criteria we use to compare the performance of the classification models. The Nath and Jones (NJ) heuristic is explained and discussed in section 5.4, where the benchmark methods that we compare it to are also presented. As is usual in academic work on proprietary information, our confidentiality agreements limit the details that can be disclosed about the sources. However, we attempt in section 5.5 to give an overview of relevant features and orders of magnitude in our databases. Section 5.6 summarizes the results of our numerical experiments. We propose some concluding remarks in section 5.7.

5.2 Problem Formulation

This section presents the details of the classification method used in our study. We first establish some notation and explain our modeling choice in the context of a brief review of LP approaches to binary classification[7]. The mathematical program is then presented in detail.

[6]Examples of case studies based on private data from several countries include [DAG96, HH96, MR99, SMH99, GT00, HA00, BCT03, CB04, FS04, ACC04].

[7]Various extensions to the multi-class case exist (see e.g., [Gri72, FG81b] and the chapters on linear discriminants in [DHS00] and [Web02]), but we will only consider binary classification models here.

5.2.1 Background and Notation

Linear discriminant functions can be thought of as hyperplanes in the feature (or characteristics) space, with a half-space assigned to each of the two classes. Let n be the number of available features, and thus the dimension of the relevant space. We denote $\mathcal{A}$ and $\mathcal{B}$ the two sets of points in $\mathbb{R}^n$ representing, respectively, the m "good" and the k "bad" training set observations. Their coordinates are represented by the matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{k\times n}$.

For a discriminating plane $P = \{x \,|w^t x = \gamma\}$, the vector w is sometimes referred to as a *scorecard* and to the product $w^t x$ as point x's *score*. Note that the often overlooked case of points lying exactly on the plane is important, and will be discussed below. We will as

A possible approach to the construction of a linear classifier is to try to minimize the number of points on the wrong side of the plane; this choice, however, results in Mixed Integer Programs that cannot currently be applied in practice to instances of industrial interest[8]. The basic idea behind most LP-based discrimination models is to find a hyperplane that minimizes some continuous, aggregate measure (such as the sum, the mean or the maximum) of the deviation of misclassified points (often called "external deviations") with respect to the plane itself. This deviation is defined as $\max(\gamma - x^t w, 0)$ for $x \in \mathcal{A}$ and $\max(x^t w - \gamma, 0)$ for $x \in \mathcal{B}$.

The most basic example of this kind of model is the minimization of the sum of deviations[9]

$$\min_{w\in\mathbb{R}^n,\gamma\in\mathbb{R}} \left\{ \sum_{i=1}^{m} \max\left\{-w^t A_i + \gamma, 0\right\} + \sum_{j=1}^{k} \max\left\{w^t B_j - \gamma, 0\right\} \right\}. \tag{5.1}$$

We recall that each row A_i or B_j represents a point in $\mathcal{A}$ or $\mathcal{B}$, respectively. Note that a point only contributes to the objective if it is misclassified, and that the objective value is zero at the optimum when the sets are linearly separable.

Some interesting variations, which we will not discuss further in this paper but are worth mentioning, include giving some weight in the objective function to correctly classified points, or "internal deviations" (see e.g., reviews in [JS90, EK90]); and minimizing deviations to class centroids, as in clustering models, instead of

[8]For a review of the literature on misclassification minimization, see chapter 3.

[9]This is also known as the *perceptron criterion*, and its origins can be traced to the work of Rosenblatt in the early 1960's [Han81, DHS00]. This is also the first model suggested in [FG81b]. Our notation follows roughly [Man99].

considering deviations to the hyperplane[10] [LO90, LCM96].

Two serious problems haunt many formulations in this family of models: null and degenerate solutions[11].

The null solution problem is the fact that $w = 0$, $\gamma = 0$ can be an optimal solution to the mathematical program. This solution is, of course, useless for discrimination purposes. Since there is a degree of freedom in the definition of plane P, a standardization constraint can in principle be imposed on w to rule out the null solution. The choice of this additional constraint, however, can be quite problematic, and this issue has permeated a large part of the literature on LP-based discrimination [Gri72, MM85, FG86, GKD88, CIS89, Koe89a, Koe89b, Glo90, Koe91, Xia93, Gle99].

Degenerate solutions are usually defined as those in which all of the points of at least one class lie on the plane itself [Koe90,Koe91,RS91]. In practice, when dealing with large datasets a large fraction, though not necessarily all, of observations from a class may fall on the plane at the optimum. This is obviously undesirable and we consider degeneracy in this broader sense of having too many training set observations with the same score.

From the classification perspective, degeneracy in LP-based discrimination is a form of overfitting, in which besides the usual detrimental effects on generalization, many training set observations remain in fact unclassifiable. This issue has perhaps not received sufficient attention in the literature and there are several empirical studies in which the effect of unclassified observations is not properly reported.

A modeling approach that has been useful in several contexts is to minimize deviations with respect to two parallel planes, one for each class, instead of a single one. We will subsequently refer to these planes as "reference planes", since each one is used as reference to measure deviation for a class. The final discriminant corresponds in fact to a plane halfway between the two (see figure 5.1).

Originally intended as an answer to the null solution problem, this family of double-plane models also alleviates somewhat mild degeneracy problems, in that at least all training set observations can be classified[12]. In fact, in double-plane

[10]This approach is equivalent to considering internal deviations on the family of double-plane models, described later in this section.

[11]Other anomalies, such as unbounded solutions, have also been studied [Koe89a] but will not be discussed here.

[12]Some double-plane models are formulated in a way that leaves an unclassifiable region between them. This is known in the literature as the "classification gap"; see [SR92] for a discussion of this issue. We focus only on models in which the reference regions for both classes overlap, thus

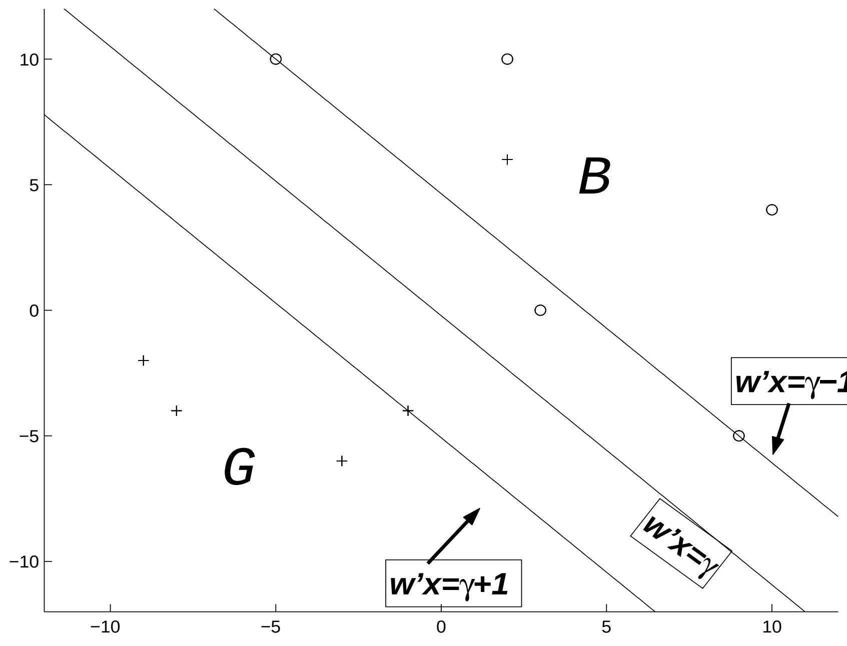

Figure 5.1: Reference planes and actual discriminant midway between them

models observations from different classes tend to stick to their respective reference planes.

The generic form of this family of programs is

$$\min_{w\in\mathbb{R}^n,\gamma\in\mathbb{R}} \left\{ \Phi_G \sum_{i=1}^{m} \max\left\{-w^t A_i + \gamma + \varepsilon, 0\right\} + \Phi_B \sum_{j=1}^{k} \max\left\{w^t B_j - \gamma + \varepsilon, 0\right\} \right\} \tag{5.2}$$

where Φ_G and Φ_B define, respectively, the weights of misclassifications from each class[13]. The parameter $\varepsilon > 0$ forces the separation between the reference planes; it is an arbitrary scaling factor and, following the usual practice, we fix it at the value of 1 for our tests, and in our subsequent discussion.

To our knowledge, the earliest double-plane model was proposed in [Smi68], with $\Phi_G = \Phi_B = \frac{1}{m+k}$. Without apparent awareness of this precedent, [Han81] presents the model with $\Phi_G = \Phi_B = 1$.

The variation proposed in [BM92], where $\Phi_G = \frac{1}{m}$ and $\Phi_B = \frac{1}{k}$, called "Robust Linear Programming" (RLP) by the authors, has become deservedly popular, and is used as building block for diverse applications ([MSW95, CM96, Man97,

leaving no gap.

[13]These weights also play a role in determining the (rather rare) cases where a null solution can emerge in a double plane model [BM92].

BFM99, DBST02]). The relevant difference with this model's predecessors is that, by weighting observations inversely to their frequency, it deals effectively with the problem of unbalanced class sizes, a crucial issue in credit scoring and other fields (see e.g., [HV03b, HV03a]).

Let us now consider the distance between the resulting reference planes in the family of double-plane models considered above. Mangasarian exposed in [Man99] a subtle link between the vector w (which is the gradient of both reference planes) and the distance that separates them, as measured by a general Lipschitz norm. It turns out that the L_p-distance between the planes is $\frac{2}{\|w\|'_p}$ where $\|\cdot\|'_p$ is the norm dual[14] to $\|\cdot\|_p$. In particular, the L_∞-distance between the planes is $\frac{2}{\|w\|_1}$ and thus, for a fixed orientation of the planes, reducing $\sum_i |w_i|$ separates them apart. Adding this term as a weighted penalty to the objective function of the RLP model, [BM00] construct a hybrid model that minimizes average deviation while maximizing the margin between the reference planes, without leaving the realm of linear programming. Furthermore, as the absolute value of the w_i coefficients is penalized in the objective function, those corresponding to uninteresting features are pushed toward zero, contributing to the task of feature selection[15]. The authors call this model "Linear Support Vector Machine" (LSVM), because it belongs to a particular family of (linear kernel) Support Vector Machines [Bur98, BC00, BMM02]. We now present the version of this model that we use for our experiments.

5.2.2 The Model

The model on which we base our tests has the form

$$\min_{w\in\mathbb{R}^n,\gamma\in\mathbb{R}} (1-\lambda)\begin{pmatrix} \Phi_A \sum_{i=1}^{m} \max\{-w^t A_i + \gamma + 1, 0\} + \\ \Phi_B \sum_{j=1}^{k} \max\{w^t B_j - \gamma + 1, 0\} \end{pmatrix} + \lambda \|w\|_1 \ . \tag{5.3}$$

As in RLP [BM92], the LSVM of [BM00] sets $\Phi_A = \frac{1}{m}$ and $\Phi_B = \frac{1}{k}$. We have found that, in dealing with large datasets, these parameters take values near or beyond the precision of common LP solvers.

We rescale the whole term by the size of the larger class (almost always $\mathcal{A}$

[14]Recall that, for $w \in \mathbb{R}^n$, $1 < p < \infty$, the L_p-norm is $\|w\|_p = \left(\sum_{i=1}^{n} |w_i|^p\right)^{\frac{1}{p}}$ and the dual norm is defined as $\|w\|'_p = \|w\|_q$ where q is such that $\frac{1}{p} + \frac{1}{q} = 1$. Also, $\|w\|'_1 = \|w\|_\infty$ and vice versa (see [Man99]).

[15]This formulation has been used in other studies, e.g., [GD03].

in credit scoring applications) and thus take $\Phi_A = 1$ and $\Phi_B = \frac{m}{k}$ This choice is numerically more robust, and still has the effect of weighting observations in inverse proportion to their class size. The original model in [BM00] has $\frac{\lambda}{2}$ instead of λ in the last term, to account for the fact that the L_∞-distance between the reference planes is $\frac{2}{\|w\|_1}$. We omit the 2 and absorb its effect in the parameter λ.

After linearizing the $\max\{\cdot\}$ and the $|\cdot|$ operators, our final formulation is[16]

$$\min_{w,\gamma,y,z} \left\{ (1-\lambda) \left(\sum_{i=1}^{m} y_i + \frac{m}{k} \sum_{j=1}^{k} z_j \right) + \lambda \sum_{r=1}^{n} s_r \right\}$$

$$\begin{aligned} &\text{subject to} && && (5.4)\\ &y_i \geq -w^t A_i + \gamma + 1 && \text{for } i = 1,..,m \\ &z_j \geq w^t B_j - \gamma + 1 && \text{for } j = 1,..,k \\ &-s_r \leq w_r \leq s_r && \text{for } r = 1,..,n \\ &y \geq 0, \quad z \geq 0, \quad s \geq 0 \end{aligned}$$

where $w \in \mathbb{R}^n$, $\gamma \in \mathbb{R}$, $y \in \mathbb{R}^m$, $z \in \mathbb{R}^k$ and $s \in \mathbb{R}^n$.

The appropriate value for the parameter λ is determined by variables such as the number of total observations, the scale of the data in matrices A and B, the degree of separability of the classes, and so on. We spanned a wide range of values for λ and kept the one that performed best in cross validation. We found that the value that best generalizes is often far from the one that provides the best fit on the training set. This suggests that there is indeed an important influence of the margin effect on generalization.

To further explore this point, we compared the performance of the LSVM model versus a basic RLP. As we report below, we found that the additional (margin) term does indeed improve considerably the generalization accuracy of the scorecard.

5.3 Assessment Criteria

In most discrimination tasks, the analyst is just interested in forecasting the class to which a new observation belongs. In some practical applications, however,

[16]Note that the variables s_r take the value $|w_r|$. This approach is equivalent to the more traditional decomposition of the variables w_r into non-negative w_r^+ and w_r^- such that $w_r = w_r^+ - w_r^-$, but facilitates a cleaner, more consistent notation. It is used, e.g., in [Man99].

an important objective is to produce a ranking of observations by predicted likelihood of belonging to a group or another. This is often the use given to scoring methods in various business processes, where managers adjust their decisions for example by considering alternative cutoff scores.

A number of common statistics measure performance across all possible cutoffs. Examples include the Mahalanobis distance and the Gini coefficient. Others, like the frequently used Kolmogorov-Smirnov (KS) statistic, consider a single cutoff which is not under the control of the analyst, with the possible risk of being irrelevant in practice [Han97, DHS00, Web02, TEC02]. The costs of potential classification errors can also be taken into account, even when uncertain [Alt80, AH99].

The criterion we adopt is the rate of "bads amongst accepted" [Han05]. We rank all observations by their score and assume they are to be considered in that order. We then plot the total accepted versus the bads accepted (and abbreviate it BAA chart). This chart is somehow similar to the ROC plot [Han97, Faw03], but does not contain the same information; one needs the prior class distribution to transform one into the other[17].

This criterion has a number of advantages:

- it relates scorecard performance to a decision variable controllable by managers, namely the percentage to accept,
- it aligns the evaluation of the scorecards to their use in the context where the data were obtained[18], and
- it allows us to average the curves along the vertical axis for cross validation purposes [Faw03] and to compare them meaningfully [Gou92].

We apply this criterion within a five-fold cross validation framework. We randomly partition sets $\mathcal{A}$ and $\mathcal{B}$ (separately) into 5 groups, and then match them arbitrarily to create 5 buckets with approximately the same class distribution as the original sample. Our scorecards are estimated on all subsets of four buckets and the BAA chart is constructed on the fifth.

When comparing a parametric family of scorecards (for instance with different number of input features) we also need summary, scalar measures. A natural one

[17]The ROC plot shows cumulative true positives on one axis and false positives on the other. ROC stands for Receiver Operating Characteristic, a term that has its origins in the use of this technique in processing noisy radio signals.

[18]The importance of aligning the assessment criterion to the planned use for the scorecard is stressed in [HV03b] and [Han05].

is to consider the areas under the BAA curves. This approach, however, is subject to the critique of considering irrelevant threshold values [Han05]. If, for example, actual acceptance rates will be fixed around 90%, the areas under the leftmost 10% of the respective BAA curves have no interest whatsoever.

We propose an alternative criterion, which is aligned to the decision making context in which our data was gathered. We assume that the relevant range of acceptance is between 70% and 90%, and propose as performance statistic the area under the BAA curve segment corresponding to this range. In reporting our results, we refer to this statistic as the *70-90 slice*.

5.4 Feature Selection

We now summarize the jackknife approach of [NJ88] and propose an interpretation and adaptation of its ideas. We then describe some alternative feature selection approaches that we compare it to.

5.4.1 The Jackknife Introduced

The procedure known in statistical pattern recognition as "the jackknife" was not originally conceived as a feature selection algorithm, but is rather a resampling technique used to estimate (and correct) biases in the error rate of the estimation of a parameter [Han97,DHS00]. The basic idea is to compare a parameter estimate computed over the whole sample with estimates obtained using various subsets of it[19]. For convenience, we transcribe verbatim in an appendix (section 5.8) Nath and Jones' own presentation of the general jackknife principle.

The NJ procedure can be summarized as follows:

1. Partition the set $\mathcal{A} \cup \mathcal{B}$ into l subsets S_i, for $i = 1, \ldots, l$.

2. Obtain l estimates of the vector $w^i \in \mathbb{R}^n$, for $i = 1, \ldots, l$ by solving the LP discrimination model corresponding to each of the sets $\bigcup_{\substack{i=1 \\ i \neq j}}^{l} S_i$, i.e., deleting one at a time each S_i from the full dataset $\mathcal{A} \cup \mathcal{B}$.

3. Compute the mean $\widehat{w}$ of the w^i vectors, and the jackknife estimate (cfr. appendix 5.8) of the standard errors, σ. Note that each component of $\sigma \in \mathbb{R}^n$

[19]The jackknife technique is related to other resampling methods, such as the bootstrap (see [ET93], ch. 3).

estimates the standard error of the mean coefficient $\widehat{w}_j$ corresponding to feature j.

4. Consider the ratio $\frac{\widehat{w}_j}{\sigma_j}$. By an argument presented in [CP77], it is supposed to follow a t-distribution with $l-1$ degrees of freedom. Retain the features for which this statistic is significant and drop the others. Alternatively, rank the features by their significance and take a certain number down the list.

5. Use the estimates $\widehat{w}_j$ corresponding to the retained features for prediction.

Note that the procedure is used simultaneously for the conceptually distinct tasks of choosing the features and estimating the corresponding coefficients $\widehat{w}_j$.

5.4.2 The Jackknife Revisited: What is really going on?

We will argue below that the general ideas behind the jackknife approach for feature selection in LP based discrimination are quite effective. However, over 15 years after the original publication of the technique, a fresh perspective on the subject is in order. In this section we discuss several issues around the jackknife approach in what, in our view, amounts to a reinterpretation of the technique. We first consider the double function of feature selection and estimation in NJ. We then interpret the approach as a compromise in a bi-objective task, for which we consider alternative formulations. Next, a discussion of the impact of unbalanced classes is presented, along with our proposal for dealing with the problem. The section ends with a summary of our version of the technique.

5.4.2.1 Feature selection vs. estimation

One should distinguish, within the original NJ framework, the combined functions of feature selection and of estimation of (presumably robust) coefficients. Instead of using the average coefficients $\widehat{w}_j$ as in NJ, one can apply the approach strictly to feature selection. The ranking produced by the NJ algorithm can be used to chose a subset of features, and then the model can be estimated for these features, from the original training dataset, as opposed to using the average of the Jackknife estimators. We compare the performance of these alternative approaches in our datasets and, as shown below, find that coefficients for the features chosen by a jackknife approach, estimated directly in the original model, outperform those obtained within NJ.

This confirms in our view the relevance of the proposed distinction.

5.4.2.2 Bicriterion interpretation

It is known that, in general, a higher (absolute) value of the resulting $\widehat{w_j}$ need not imply that the feature in question is more interesting (e.g., [Han97]). One reason for this can be for example, that the features have very different scales. In a sensible scorecard the coefficients would adjust for this fact, the absolute value of those corresponding to features with larger scales being relatively smaller. We argue, however, that if the features have been rescaled to the same range, as is the case in common practice (and in our datasets), a ranking of features by absolute value of the corresponding $\widehat{w_j}$ may be intuitively appealing.

We compare, for our datasets, the performance of a classifier estimated on the full original set of features with one that only takes, for example, the first 100 ranked by $|\widehat{w_j}|$, and confirm that the latter is superior.

On the other hand, consider the various coefficients w_j^i obtained in the NJ procedure for each feature $j = 1, \ldots, n$ when each subset S_i is removed from the training set, for $i = 1, \ldots, l$. To the extent that l is relatively large, the removal of the subsets S_i should not dramatically change the structure of the problem. It is thus intuitively plausible that the w_j^i corresponding to features with highest predictive power should have relatively small variation.

One would like therefore to choose features with higher $|\widehat{w}_j|$ and lower σ_j. A possible way to deal with this double criterion is to consider the ratio of the two components; this is exactly what NJ does. In fact, the t-statistic is just a monotonic transformation of $\frac{|\widehat{w}_j|}{\sigma_j}$. The experiments presented below show that, for our datasets, this treatment of the bicriterion challenge outperforms the simple ranking by $|\widehat{w}_j|$.

An alternative way of dealing with the two criteria is to aggregate them linearly and then rank them, in a function of the form $\alpha\,|\widehat{w}_j| + \beta\sigma_j$. We explore the consequences of this approach on our databases, and find that the performance is extremely similar to that of the NJ criterion. This result further supports our interpretation of the NJ method as a way of dealing with a bicriterion compromise.

Consider a feature such that its $\widehat{w}_j$ are very close to zero. The ratio $\frac{|\widehat{w}_j|}{\sigma_j}$ might, nevertheless, be relatively large even if the feature is uninteresting. On the other hand, one would tend to consider a coefficient that changes signs to be associated with a bad candidate feature. The view that excessive variability is undesirable and large absolute values are desirable suggests a heuristic rule to deal with this issues: remove all variables whose mean coefficient estimates have absolute values too close to the precision of the solver, and also all variables whose coefficients change signs. Our tests show this approach to be clearly beneficial, at least for the case of our datasets.

Despite the common sense appeal of the bicriterion interpretation presented, as well as its empirical validation on our dataset examples, it should be noted that the problem at hand does not conform to the classic framework for the analysis of multi-criteria decision-making (e.g., [KR76]). For example, the usual domination relationships do not appear to apply.

5.4.2.3 Unbalanced classes

In many applications of practical interest the classes are unbalanced, in the sense that many more observations exist of one than of the other. This is indeed the case in credit scoring, and this issue has important implications for both the construction and the assessment of classification models [Han05, HV03a, HV03b]. In the context of the jackknife approach, if the classes are unbalanced, the random partition of the original training data into the S_i subsets can lead to some of the l models being estimated on sets with dramatically different class proportions to the target population priors. This will introduce a dimension of variability to the estimates of w_j that is unrelated to the relative predictive contribution of the features. As mentioned above, the way we deal with this problem in our implementations is to partition the sets $\mathcal{A}$ and $\mathcal{B}$ separately, and then randomly match pairs to create the S_i subsets.

5.4.2.4 Our approach summarized

In summary, our proposed adaptation of the jackknife approach to feature selection in LP based discrimination models has the following distinctive characteristics:

- Partition the original training set in a way that preserves the relative proportions of the classes.
- Remove variables for which the coefficients w_j^i change sign for $i = 1, \ldots, l$, as well as those with extremely small $|\widehat{w}_j|$.
- Rank the variables by $\frac{|\widehat{w}_j|}{\sigma_j}$ or other summary function of mean and dispersion[20] of w_j^i.

[20]The ranking obtained by $\frac{|\widehat{w}_j|}{\sigma_j}$ will of course be the same as that of the t-statistic, but we

- Choose the number of variables to keep by observing the generalization performance of a range of values, or by exogenous, ad-hoc domain considerations[21].
- Keep the selected subset of variables and use them to estimate the model on the full training set (as opposed to using the jackknife estimates).

5.4.3 A Filter Benchmark

Considered only as a feature selection algorithm, the jackknife approach as discussed above can be argued to belong to the family of wrappers, i.e., the features are chosen considering explicitly the model that is ultimately used for prediction (in this case, the LP program itself)[22].

Consider a set of features selected by a wrapper method, built around a given classifier. By construction, these features should be expected to perform better than a set of features chosen by a filter method, when used with the classifier in question. It would therefore not be surprising that the predictive accuracy of our LP model, with the features selected with the jackknife approach, be higher than if a subset of features chosen with a filter were used. We find, however, interesting to study to what extent this expectation is confirmed in our datasets.

A recent paper [LS05] compares the performance of several feature selection methods on a credit scoring problem, using the publicly available WEKA package[23] We tried five common filter algorithms implemented in this software to rank the features of our databases, and considered the best 100 of each list. We then tried our LP model with these feature sets, comparing their cross validation performance.

The algorithms considered are *ChiSquaredAttributeEval*, *GainRatioAttributeEval*, *InfoGainAttributeEval*, *ReliefFAttributeEval* and *SymmetricalUncertAttributeEval*. They are described in detail in [WF05]. We found that the χ^2-criterion

stress the intuitive interpretation of the trade-off between $|\widehat{w}_j|$ and σ_j.

[21]Our point here is that defining some "significance threshold" and keeping variables that are below it is meaningless from the bicriterion point of view.

[22]There is a slight conceptual abuse in this classification, because in the original definition of a wrapper in [JKP94] and [KJ97] the combinatorial space of features is searched as subsets of features are assessed by repeatedly querying the prediction algorithm as a "black box" or oracle. In the case at hand, the resampling approach produces a ranking of features by repeatedly computing the coefficients that correspond to *all* features. The space of subsets of features is thus not explicitly explored. However, since the ranking is produced by repeatedly invoking the induction algorithm itself, we believe that characterizing the jackknife approach as a wrapper is appropriate for our discussion.

[23]Available at `http://www.cs.waikato.ac.nz/ml/weka/` and discussed in the companion book [WF05].

of *ChiSquaredAttributeEval* outperformed by far the others; it is thus the one we present in comparison to the Jackknife methods discussed above.

5.5 The Databases

Our databases were provided by two Latin American lenders, and are believed to be representative of their portfolios[24]. The one to which we refer as the *auto* problem corresponds to a car-loan portfolio, and we call the *home* problem a mortgage database. As is usually the case in Credit Scoring tasks, the features are mostly applicant demographic information and characteristics of the loan contract.

Both datasets were processed according to common industry practices; our work focuses on the final, classification task, and all of the results reported were computed on the same input datasets. A few comments, however, are in order with respect to the construction of the input columns.

All categorical and numerical fields were expanded into binary variables. This transformation, common among practitioners, appears to have at least three advantages. First, the discretization of numerical variables allows the linear discriminant to capture potential non-linear effects. For instance, the effect of age on credit performance is sometimes seen to be increasing for young applicants and decreasing for the oldest groups. Second, the creation of new variables to capture cross-effects between characteristics becomes straightforward; and third, as argued below in the context of feature selection for LP-based discriminants, the fact that all variables are binary facilitates the task of comparing and interpreting the resulting coefficients.

Following a common industry practice, the discretization of continuous variables[25] was performed heuristically using the log-likelihood (or "weight of evidence") of the resulting variables as guideline ([TEC02]). Although some more formal approaches to this task have been proposed (e.g., [HA00,Gle04]), their practical applicability in industrial-size databases has not been documented.

In order to capture some (non-linear) cross effects between variables, additional columns were generated by multiplying pairs of variables from different groups (e.g., age vs. marital status). Some knowledge discovery techniques that also combine binary variables do so in an exhaustive or systematic manner (e.g., [ABH$^+$03]). In

[24]In compliance with our confidentiality agreements, some of the information in this section is deliberately vague.

[25]This process is known as "coarse classification of characteristics" in the Credit Scoring jargon.

our case, as is usual in Credit Scoring practice, the cross products considered are chosen somewhat arbitrarily, guided by domain knowledge.

The class label was constructed according to the performance of the loan over a given period of time. For instance, on the *auto* database, the “bad” observations correspond to loans that have been past due for over 60 days. This is unlike some other empirical work in the credit classification literature, which is based on databases where class belonging is determined by expert opinion, rather than actual borrower behavior (e.g., [LS05, BKU02]).

The final *auto* database contains 19,657 observations in 208 dimensions, with about 4% of “bads”, while the *home* database consists of 10,579 observations in 673 dimensions, with about 17% of “bads”.

5.6 Experimental Results

We now summarize the main results of our numerical experiments on the *auto* and *home* databases. Although our databases are somehow independent, in that they represent different product segments, come from different institutions and where preprocessed somehow differently, the usual caveats with respect to any generalization of our findings apply.

Unless otherwise indicated, the references to the jackknife procedure refer to our variation of it, as summarized in ¶5.4.2.4.

5.6.1 Structure of LP model: including a margin term

In figures 5.2 and 5.3 we compare the performance of our implementation of the LSVM to the baseline RLP model. Although the effect is less clearcut for the *auto* than for the *home* database, we believe there is some evidence that the addition of the margin term to the objective function indeed improves generalization.

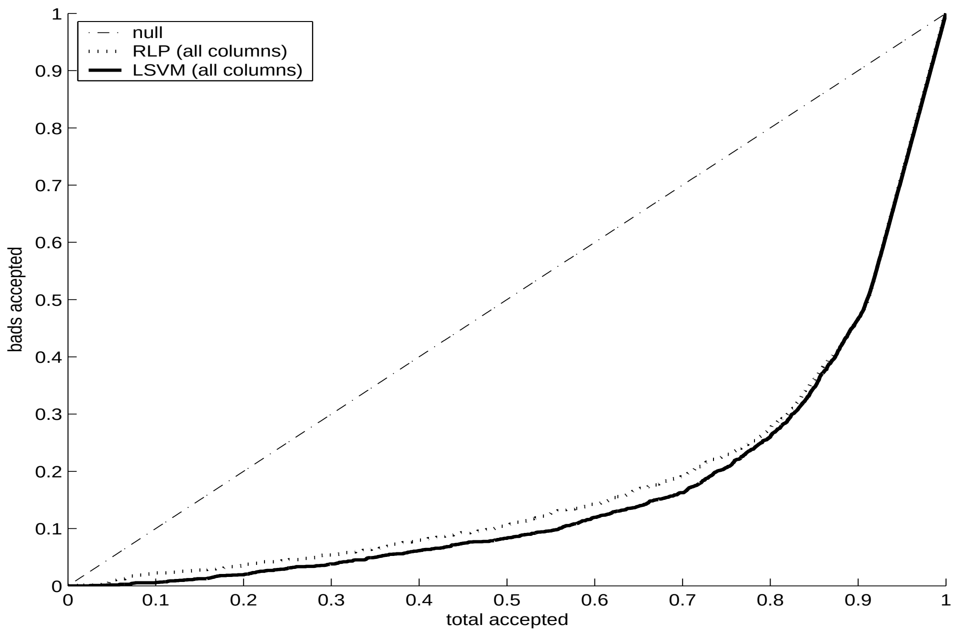

Figure 5.2: RLP vs. LSVM in the *home* problem

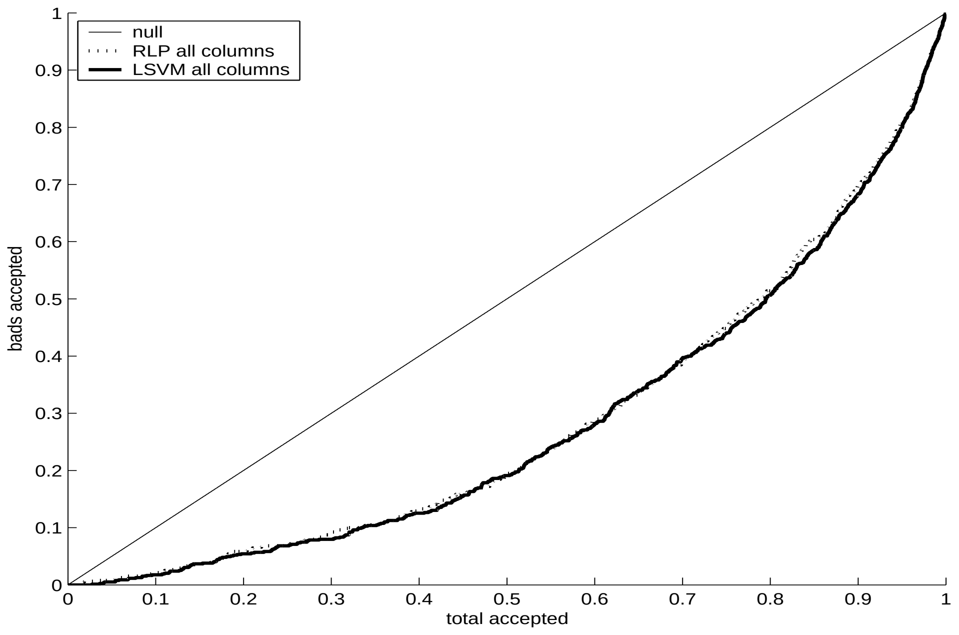

Figure 5.3: RLP vs. LSVM in the *auto* problem

5.6.2 Removal of obviously bad candidates

Figure 5.4 compares the performance of a model that includes all the features to one that is estimated after having removed all the variables that either change sign or have systematically small values in the various jackknife estimates. We believe there is some evidence that this step does improve performance.

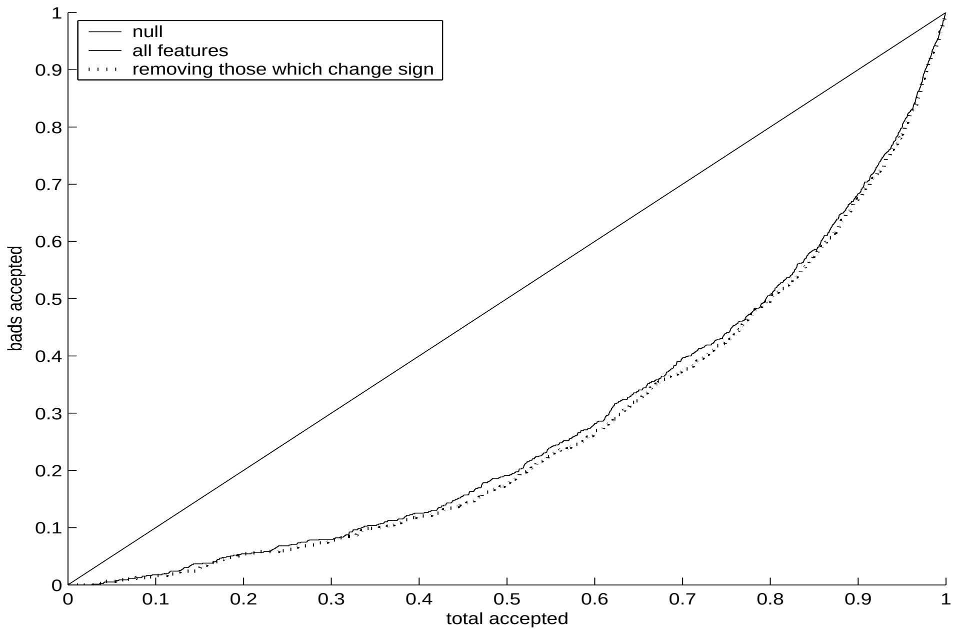

Figure 5.4: Effect of removing coefficients that change sign or are always extremely small: *auto* dataset.

5.6.3 Source of coefficient estimates

We compare the performance of the coefficients obtained under the criterion used in the original NJ setting (i.e., the proper jackknife estimates), to those obtained under our proposed approach of using the jackknife only as a pure feature selection procedure and then estimating the parameters on the full training set.

Figures 5.5 and 5.6 show the results for both sets. Although the difference appears to be neglegible for the *auto* data, we find support for our preference for these coefficients in the *home* example.

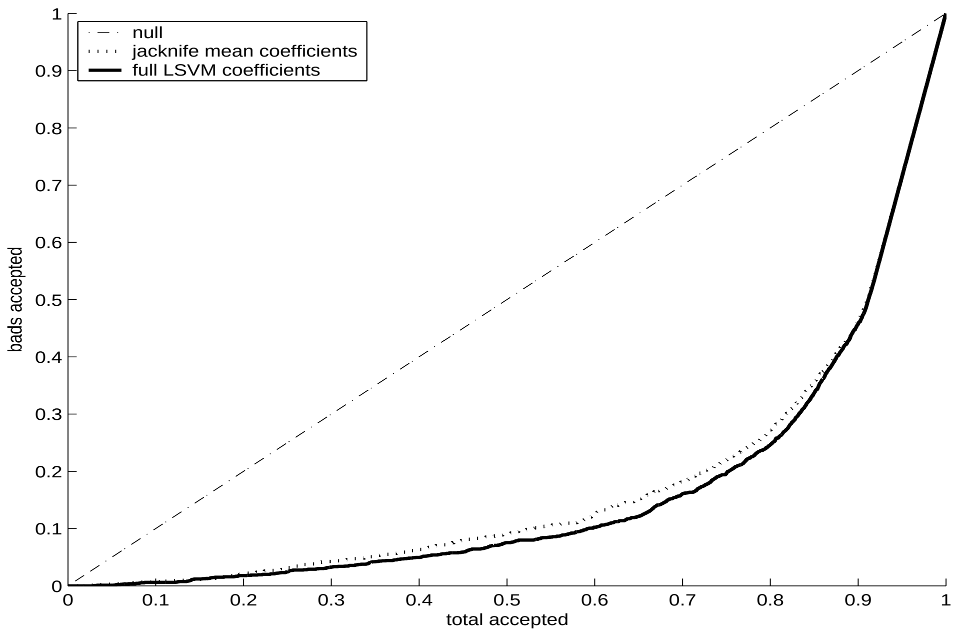

Figure 5.5: Jackknife vs. LSVM coefficients in the *home* problem

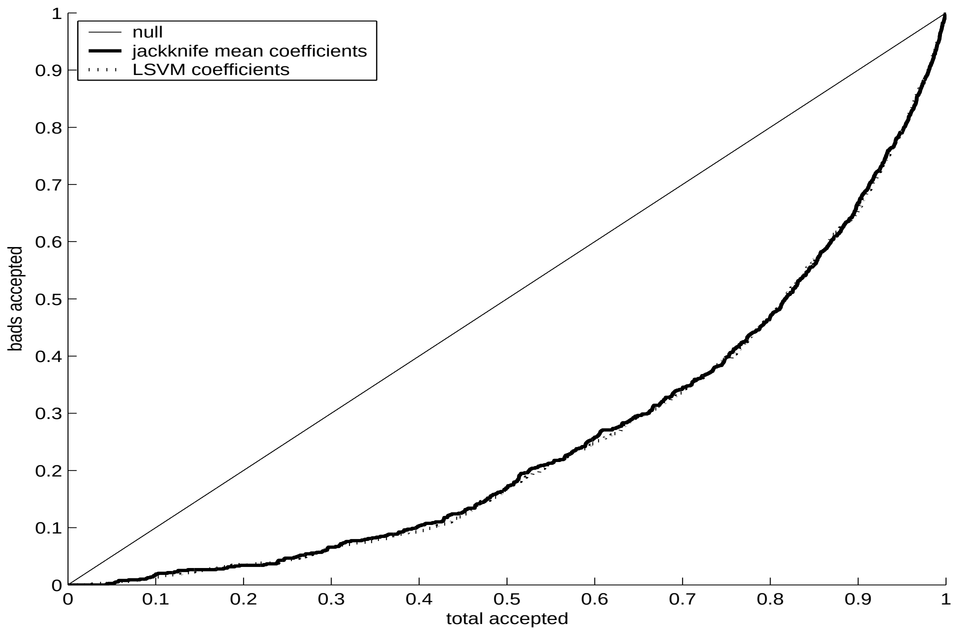

Figure 5.6: Jackknife vs. LSVM coefficients in the *auto* problem

5.6.4 Superiority to filters

We compared our LP model using the best 100 variables as ranked by the WEKA χ^2 filter, to the best 100 as ranked by the baseline jackknife. Figure 5.7 shows the result for the *auto* problem; the *home* database produced a similar plot.

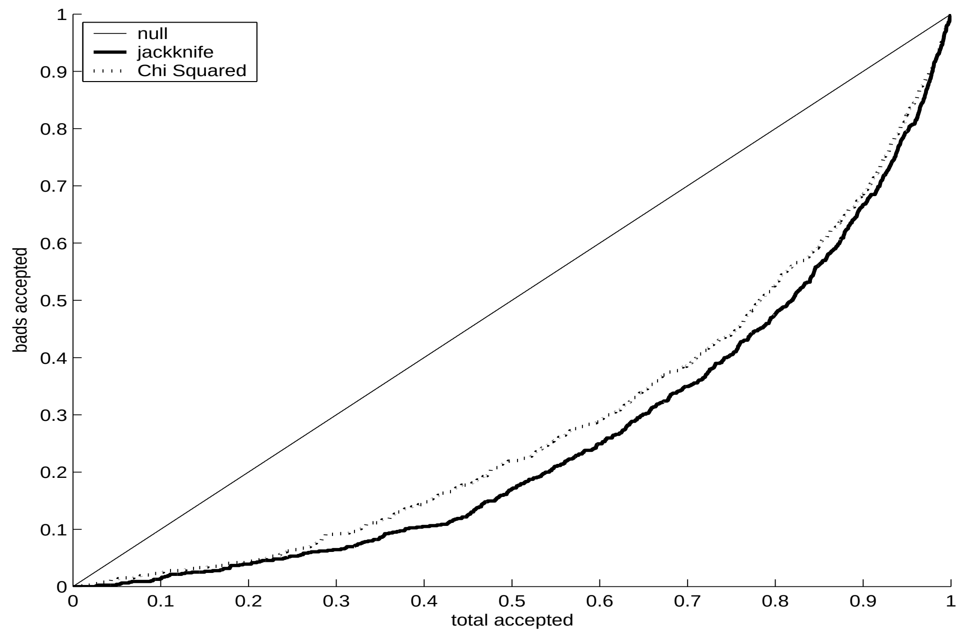

Figure 5.7: Features selected by the jacknife vs. χ^2-criterion in the *auto* problem

As mentioned above, this was the best performing filter in the set of five common algorithms tested. We confirm that a jackknife approach, perhaps unsurprisingly, easily outperforms it.

5.6.5 Single versus double criterion

Recall that the jackknife method produces l estimates for each of the scorecard coefficients w_j, and that these estimates are used to rank features, as discussed in ¶5.4.2. In figure 5.8 we compare the performance of three approaches to rank features:

- *absmeans*: Decreasing absolute value of the mean $\widehat{w_j^i}$ of the jackknife estimates
- *absmeans/std*: Decreasing ratio of the absolute value of the mean of the jackknife estimates to their standard deviation, i.e., $\frac{|\widehat{w}_j|}{\sigma_j}$

- *linear*: Decreasing value of a linear combination[26], $\alpha\,|\widehat{w}_j| + \beta\sigma_j$, of the absolute value of the mean $\widehat{w^i_j}$ of the jackknife estimates and their standard deviation σ_j, i.e., the alternative way to deal with the bicriterion problem discussed in ¶5.4.2.2

The horizontal axis shows the number of variables retained, ranging from 40 to 300, from the ordered list produced by each criterion. The vertical axis measures the *slice70-90* statistic (cfr. section 5.3) obtained for each method. Recall that a lower value corresponds to a better classifier.

The ranking obtained by considering only the means is clearly dominated by the others. Also note that, although the NJ criterion does better than the alternative linear formulation for 40 variables, the performance is virtually identical beyond 60 variables.

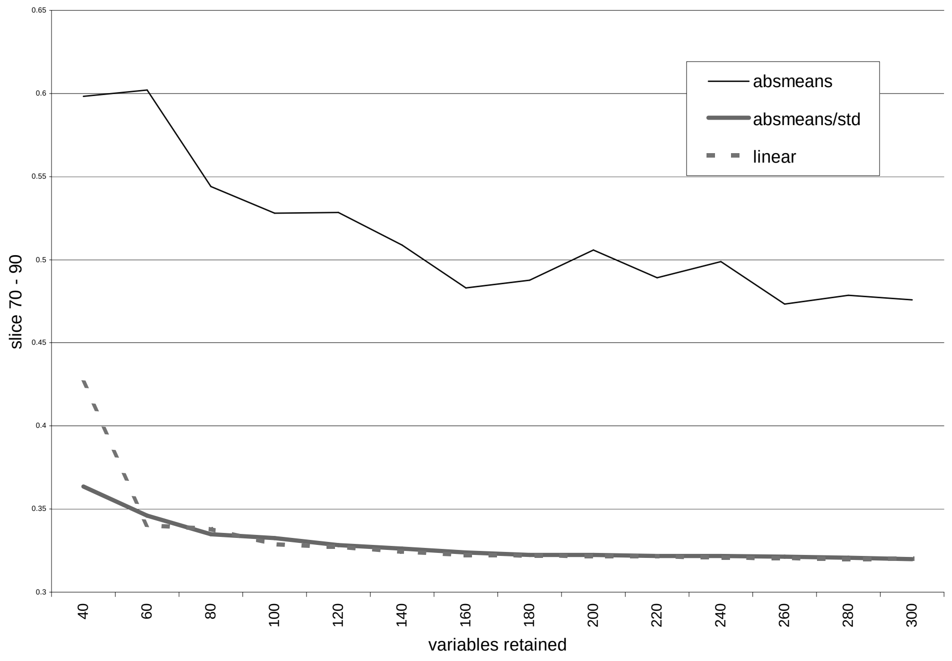

Figure 5.8: *slice70-90* as function of number of variables retained (*home* problem)

5.7 Conclusions

We explore the problem of feature selection in a linear programming approach to building a scorecard. We report numerical experiments on two real credit databases.

[26]We set the arbitrary scaling parameters $\alpha = 1$ and $\beta = .25$

The model we use is the Linear Support Vector Machine (LSVM). We find LSVM's performance to be superior (for our data) to that of the model known as Robust Linear Programming, its predecessor within the family of double-plane classifiers.

We revisit an approach introduced by Nath and Jones in 1988, based on the jackknife principle. We propose some improvements in the implementation of their ideas, and show their potential to improve performance:

- remove variables that change sign or have average values near the solver's precision,
- preserve class proportions in the jackknife samples, and
- use the jackknife strictly as a feature selection method, and estimate independently the scorecard coefficients (as opposed to using the jackknife estimates).

We offer an intuitive reinterpretation of the ideas behind the jackknife procedure, exposing it as the solution to a bicriterion problem: the jackknife estimates of interesting variables should have higher absolute mean values and lower variance.

This interpretation suggests the idea of exploring other heuristics to rank features. We try a linear combination of mean absolute value and variance, and show that it performs almost identically to the original jackknife procedure.

5.8 Appendix: The general jackknife principle

We here transcribe verbatim an excerpt from the appendix of [NJ88], where the general principle of the jackknife on which they base their method is presented:

> Suppose a random sample of size n is obtained by observing the variable X on each of the sampled units yielding $X_1, X_2, \ldots, X_n$. Partition the sample into k (k is selected by the researcher) subsets of size M_i ($i = 1, 2, ..., k$) and denote the subsets $S_1, S_2, \ldots, S_k$. Suppose $\overline{\theta'}$ is an estimate of the parameter θ using the complete sample and θ'_i is the estimate obtained after the ith subset S_i of size M_i is deleted from the complete sample. The jackknife estimate of the standard error of θ' is:
>
> $$S_{\theta'} = \left[\left(\frac{k-1}{k}\right)\sum_{i=1}^{k}\left(\theta'_i - \overline{\theta'}\right)^2\right]^{\frac{1}{2}}$$
>
> where
>
> $$\overline{\theta'} = \frac{1}{k}\sum_{i=1}^{k}\theta'_i$$
>
> is the mean of the k estimates obtained by deleting one subset at a time. The estimate $S_{\theta'}$ is known to be a robust estimate of the true standard error of θ'.

Note the subtle differences in notation with respect to the conventions adopted in our exposition. For example, we reserve n for the dimension of the space of the problem (and use l for the number of subsets in the partition of the dataset), and m and k for the cardinality of sets $\mathcal{A}$ and $\mathcal{B}$. To avoid confusion with the sets S_i, we refer to the jackknife estimate of the standard error as σ instead of $S_{\theta'}$.

CONCLUSION

Linear discriminants are an important pillar in the theory and practice of automatic classification. Since their appearance in academic literature almost 60 years ago, a considerable corpus of research has focused on them from the perspectives of various disciplines, including statistics, computer science and operations research. This thesis approaches some issues in binary linear discrimination with the application of two optimization techniques: a modern metaheuristic framework and classical linear programming.

It is common to visualize the discriminant as a hyperplane in the Euclidean space of features, and the training set containing the two classes of objects to be separated as points in this space. As in most cases of practical interest the classes are not linearly separable, the hyperplane is chosen to minimize some error criterion.

The first such criterion we consider is the minimization of the sum of L_p-norm distances of misclassified points to the plane. For truly arbitrary, integer or fractional, values of p, no general exact solution is known. We successfully apply the Variable Neighborhood Search (VNS) heuristic framework to this problem. The solutions found are reasonably accurate, and scale very well to large problems. We explore, on a set of real-life, publicly available databases, the generalization consequences of the choice of p. We find that the choice of the norm does matter, but in a case dependent way. The practical implication of this result is that, if for whatever reason L_p-norm minimization is the criterion of choice for a linear discrimination task, a range of values of p should be considered.

Exact solutions can be obtained in the case of the "Manhattan" ($p = 1$), Euclidean ($p = 2$) and Max ($p = \infty$) distances. We present a Mixed Integer Programming (MIP) formulation for the L_∞-norm and explore empirically the use of the bounds obtained by our VNS implementation to accelerate exact solutions of L_2-norm and L_∞-norm problems, both of which are difficult in large instances. We conclude that the use of our bounds is not worthwhile in small problems that can be solved exactly in little time, but that very significant time savings can be achieved in larger instances.

The next criterion we consider is the minimization of the number of misclassified points. The exact solution of this problem leads to large MIP models that are often difficult to solve. We propose and test an improvement to a classic formulation, i.e., provide tighter constraints, and conclude that, for a number of problems, it reduces significantly the exact solution time.

We adapt and implement the VNS framework to find heuristic solutions for the misclassification minimization problem. We find that it is fast, and that the solutions it finds generalize reasonably well as compared with some other linear classifiers.

An important family of linear discrimination models adopts as criterion the minimization of a measure of deviation of misclassified points to the plane, which are not necessarily distances. The final part of the thesis explores the issue of feature selection in the context of a model belonging to this family. Our work is centered on a case study of two real-life credit databases.

We study a feature selection method based on the jackknife principle, which was introduced to the literature over 15 years ago specifically in the context of the construction of discriminants by linear programming. We suggest and test some improvements to this method, and conclude that they are useful, at least for the case of our datasets.

We also propose a reinterpretation of this method as implicitly dealing with a bicriterion problem, and show that consideration of only one of the two criteria in question dramatically reduces the classification performance. Our interpretation suggests alternative formulations, and a potential for wider application of the underlying ideas.

A number of avenues for further research can be derived from the work reported in this thesis. Some salient examples are:

- Improving the performance of the VNS implementation, e.g., by adapting alternative local descent modules, more suitable for problems in relatively higher dimensions;

- Applying the heuristic bounds and solutions to the acceleration of exact solution of the misclassification minimization problem; and,

- Exploring alternative approaches to the implicit bicriterion challenge in the jackknife approach to feature selection

APPENDIX: DATA SET DETAILS

This appendix describes the databases used in chapters 1, 2 and 3. We considered the instances from the UCI Machine Learning Repository [DNM98] 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.

Pima refers to the Pima Indians Diabetes database.

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.

Musicant's NDC generator [Mus98] 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.

The Sym2k and Sym6d series, used for larger L_∞-norm instances, were constructed by fixing arbitrary centers, assigning one half of the points to each of them (Sym stands for symmetric) and generating independent columns from a normal distribution for each of them. The first center was fixed at the origin and the second one was set along the ray defined by a vector of ones of the appropriate dimension, at a distance adjusted as to keep the L_1-norm full set fit at about 89%. This criterion to approximate and control the difficulty of the problem was used because of the relative ease of obtaining the exact L_1-norm solution.

Unless otherwise noted, all databases were linearly standardized to the range $[0, 1]$.

The UCI files are available at:

http://www.hec.ca/pages/alejandro.karam/data

The NDC and SYM files are rather large, but are available upon request to alejandro.karam@hec.ca.

REFERENCES

[ABH$^+$03] S. Alexe, E. Blackstone, P. Hammer, H. Ishwaran, M. Lauer, and C. Pthier Snader. Coronary risk prediction by logical analysis of data. *Annals of Operations Research*, 119(1):15–42, 2003.

[AC01] T. Astebro and G. Chen. The economic value of reject inference in credit scoring. In L. Thomas, J. Crook, and D. B. Edelman, editors, *Credit Scoring and Credit Control VII*, Edinburgh, Scotland, 2001.

[ACC04] R. B. Avery, P. S. Calem, and G. B. Canner. Consumer credit scoring: Do situational circumstances matter? *Journal of Banking & Finance*, 28(4):835–856, 2004.

[AH99] N. M. Adams and D. J. Hand. Comparing classifiers when the misallocation costs are uncertain. *Pattern Recognition*, 32:1139–1147, 1999.

[AHJS00] C. Audet, P. Hansen, B. Jaumard, and G. Savard. A branch and cut algorithm for nonconvex quadratically constrained quadratic programming. *Math. Program.*, 87(1, Ser. A):131–152, 2000.

[AHK$^+$04] C. Audet, P. Hansen, A. Karam, C.T. Ng, and S Perron. Exact solution of l_∞-norm and l_2-norm plane separation. Les Cahiers du GERAD G-2004-84, Groupe d'études et de recherche en analyse des décisions, november 2004.

[AHT01] N. M. Adams, D. J. Hand, and R. Till. Mining for classes and patterns in behavioural data. *Journal of the Operational Research Society*, 52(9):1017–1024, 2001. SEP.

[Alt68] E.I. Altman. Financial ratios, discriminant analysis, and the prediction of corporate bankruptcy. *Journal of Finance*, 23(4):589–609, 1968.

[Alt80] E.I. Altman. Commercial bank lending: Process, credit scoring and lending error costs. *Journal of Financial and Quantitative Analysis*, 15(4):813–832, 1980.

[BC00] K. Bennett and C. Campbell. Support vector machines: hype or hallelujah? *SIGKDD Explor. Newsl.*, 2(2):1–13, 2000.

[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 salesman problem. Technical report, University of Nottingham, 1999.

[BCT03] J. Banasik, J. Crook, and L. Thomas. Sample selection bias in credit scoring models. *Journal of the Operational Research Society*, 54(8):822–832, 2003.

[BFM99] P. S. Bradley, U. M. Fayyad, and O. L. Mangasarian. Mathematical programming for data mining: Formulations and challenges. *INFORMS Journal on Computing*, 11(3):217–239, 1999. TY - JOUR.

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

[BHM02] N. Belacel, P. Hansen, and N. Mladenović. Fuzzy J-means: a new heuristic for fuzzy clustering. *Pattern Recognition*, 35(10):2193–2200, 2002.

[BKU02] V. Bugera, H. Konno, and S. Uryasev. Credit cards scoring with quadratic utility functions. *Journal of Multi-Criteria Decision Analysis*, 11(4-5):197–211, 2002.

[BLSvW96] B. Back, T. Laitinen, K. Sere, and M. van Wezel. Choosing bankruptcy predictors using discriminant analysis, logit analysis, and genetic algorithms. Technical Report TUCS-TR-40, Turku Centre for Computer Science, 16, 1996.

[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.

[BM00] P. S. Bradley and O. L. Mangasarian. Massive data discrimination via linear support vector machines. *Optimization Methods & Software*, 13(1):1–10, 2000.

[BMM02] P. S. Bradley, O. L. Mangasarian, and D. R. Musicant. Optimization methods in massive datasets. In J Abello, P M Pardalos, and M G Resende, editors, *Handbook of Massive Datasets*, pages 439–472. Kluwer Academic Publishers, 2002.

[Bur98] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[BVGV$^+$03] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, 54(6):627–635, 2003.

[CB04] J. Crook and J. Banasik. Does reject inference really improve the performance of application scoring models? *Journal of Banking & Finance*, 28(4):857–874, 2004. APR.

[CH00] G. Caporossi and P. Hansen. Variable neighborhood search for extremal graphs 1: The AutoGraphiX system. *Discrete Mathematics*, 212:29–44, 2000.

[Cha64] A. Charnes. Some fundamental theorems of perceptron theory and their geometry. In J. T. J. T. Tou and R. H. Wilcox, editors, *Computer and Information Sciences*, pages 67–74. Spartan Books, Washington, D. C., 1964.

[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.

[CP77] M. R. Crask and W. D. Perreault. Validation of discriminant analysis in market research. *JMR, Journal of Marketing Research (pre-1986)*, 14(000001):60–68, 1977.

[DAG96] G. Dionne, M. Artís, and M. Guillén. Count data models for a credit scoring system. *Journal of Empirical Finance*, 3:303–325, 1996.

[DBST02] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

[DDS02] G. Desaulniers, J. Desrosiers, and M.M. Solomon. Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In *Essays and surveys in metaheuristics (Angra dos Reis, 1999)*, Oper. Res./Comput. Sci. Interfaces Ser., pages 309–324. Kluwer Acad. Publ., Boston, MA, 2002.

[DEG92] R.H. Davis, D. B. Edelman, and A. J. Gammerman. Machine-learning algorithms for credit-card applications. *Journal of Mathematics Applied in Business & Industry,*, 4:43–51, 1992.

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2000.

[DMS01] Z. Drezner, G. A. Marcoulides, and M. H. Stohs. Financial applications of a tabu search variable selection model. *Journal of Applied Mathematics & Decision Sciences*, 5(4):215–234, 2001.

[DNM98] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[EK90] S. S. Erenguc and G. J. Koehler. Survey of mathematical-programming models and experimental results for linear discriminant-analysis. *Managerial and Decision Economics*, 11(4):215–225, 1990.

[ET93] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.

[Faw03] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4, HP Intelligent Enterprise Technologies Laboratory, January 2003.

[FG81a] N. Freed and F. Glover. A linear programming approach to the discriminant problem. *Decision Sciences*, 12(1):68, 1981.

[FG81b] N. Freed and F. Glover. Simple but powerful goal programming models for discriminant problems. *European Journal of Operational Research*, 7(1):44, 1981.

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

[FI92] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[Fis36] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(II):179–188, 1936.

[FM01] G. Fung and O.L. Mangasarian. Proximal support vector machine classifiers. In *Knowledge Discovery and Data Mining*, pages 77–86, 2001.

[FS04] D. P. Foster and R. A. Stine. Variable selection in data mining: Building a predictive model for bankruptcy. *Journal of the American Statistical Association*, 99(466):303–313, 2004. JUN.

[GD03] G. Guo and C. Dyer. Simultaneous feature selection and classifier training via linear programming: A case study for face expression recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, volume I, pages 346–352, 2003.

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

[GKD88] F. Glover, S. Keene, and B. 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.

[Gle04] J. J. Glen. Dichotomous categorical variable formation in mathematical programming discriminant analysis models. *Naval Research Logistics*, 51(4):575–596, 2004.

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

[Gou92] C. Gourieroux. Courbes de performance, de sélection et de discrimination. *Annales d'Économie et de Statistiques*, 28:107–123, 1992.

[Gri72] R. Grinold. Mathematical programming methods of pattern classification. *Management Science*, 3:272–290, 1972.

[GT00] J. Galindo and P. Tamayo. Credit risk assessment using statistical and machine learning: Basic methodology and risk modeling applications. *Computational Economics*, 15:107–143, 2000.

[GTMM03] M. García-Torres, J. A. Moreno Pérez, and J. Marcos Moreno Vega. Vns para clasificación supervisada. In *MAEB 2003*, pages 343–352, Gijón, Spain, 2003.

[HA00] D. J. Hand and N. M. Adams. Defining attributes for scorecard construction in credit scoring. *Journal of Applied Statistics*, 27(5):527–540, 2000.

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

[Han97] D. J. Hand. *Construction and assessment of classification rules.* John Wiley & Sons, 1997.

[Han01] D. J. Hand. Modelling consumer credit risk. *IMA Journal of Management Mathematics*, 12:139–155, 2001.

[Han05] D. J. Hand. Good practice in retail credit scorecard assessment. *Journal of the Operational Research Society*, S(advance online publication):1–9, 2005.

[Hay03] E. Hayden. Are credit scoring models sensitive with respect to default definitions? evidence from the austrian market. In *EFMA 2003 Helsinki Meetings*, Helsinki, 2003.

[HBUM03] P. Hansen, J. Brimberg, D. Urosevic, and N. Mladenovic. Primal-dual variable neighborhood search for bounded heuristic and exact solution of the simple plant location problem. Les Cahiers du GERAD G-2003-64, Groupe d'études et de recherche en analyse des décisions, october 2003.

[HH96] W. E. Henley and D. J. Hand. A k-nearest-neighbour classifier for assessing consumer credit risk. *Statistician*, 45(1):77–95, 1996.

[HH97] D. J. Hand and W. E. Henley. Statistical classification methods in consumer credit scoring: a review. *Journal of the Royal Statistical Society Series a-Statistics in Society*, 160:523–541, 1997. Part 3.

[HJM98] P. Hansen, B. Jaumard, and N. Mladenović. Minimum sum of squares clustering in a low dimensional space. *Journal of Classification*, 15:37–56, 1998.

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

[HM99] P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In S. Voss et al., editor, *Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer, 1999.

[HM01a] P. Hansen and N. Mladenović. J-means: A new local search heuristic for minimum sum-of-squares clustering. *Pattern Recognition*, 34(2):405–413, 2001.

[HM01b] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[HM01c] P. Hansen and N. Mladenović. Variable neighbourhood search. In P. Pardalos and M. Resende, editors, *Handbook of Applied Optimization*, chapter 3.6.9, pages 221–234. Oxford University Press, 2001.

[HMU04] P. Hansen, N. Mladenović, and D. Urošević. Variable neighborhood search for the maximum clique. *Discrete Applied Mathematics*, 145(1):117–125, 2004.

[Hua98] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[HV03a] D. J. Hand and V. Vinciotti. Choosing k for two-class nearest neighbour classifiers with unbalanced classes. *Pattern Recognition Letters*, 24(9-10):1555–1562, 2003. JUN.

[HV03b] D. J. Hand and V. Vinciotti. Scorecard construction with unbalanced class sizes. *Journal of the Iranian Statistical Society*, 2(2):189 – 205, 2003.

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

[JKP94] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994.

[Joa94] D. N. Joanes. Reject inference applied to logistic regression for credit scoring. *Journal of Mathematics Applied in Business & Industry*, 5:35–43, 1994.

[JS90] E. A. Joachimsthaler and A. Stam. Mathematical programming approaches for the classification problem in two-group discriminant analysis. *Multivariate Behavioral Research*, 25(4):427–454, 1990.

[KE90] G. J. Koehler and S. S. 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.

[KJ97] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

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

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

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

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

[KR76] R.L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[KS04] Y. S. Kim and S. Y. Sohn. Managing loan customers using misclassification patterns of credit scoring model. *Expert Systems with Applications*, 26(4):567–573, 2004. MAY.

[LCM96] 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. *European Journal of Operational Research*, 88(2):358–367, 1996.

[Liu02] Y. Liu. A framework of data mining application process for credit scoring. Technical report 01/2002, Institut für Wirtschaftsinformatik, Universität Göttingen, 2002.

[LO90] C. K. Lee and J. K. Ord. Discriminant analysis using least absolute deviations. *Decision Sciences*, 21(1):86, 1990.

[LS05] Y. Liu and M. Schumann. Data mining feature selection for credit scoring. *Journal of the Operational Research Society*, S(advance online publication):1–10, 2005.

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

[Man65] O.L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

[Man97] O.L. Mangasarian. Mathematical programming in data mining. *Data Mining and Knowledge Discovery*, 1(2):183–201, 1997.

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

[Mel97] E. Melachrinoudis. An analytical solution to the minimum $\mathcal{L}_p$-norm of a hyperplane. *Journal of Mathematical Analysis and Applications*, 211:172–179, 1997.

[MH97] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100, 1997.

[Min61] R. Minnick. Linear-input logic. *IRE Transaction on Electronic Computers*, EC-10:6–16, 1961.

[MJPČ03] N. Mladenović, V. Kovačević-Vujčić J. Petrović, and M. Čangalović. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operations Research*, 151:389–399, 2003.

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

[MM00] O.L. Mangasarian and D.R. Musicant. Active support vector machine classification. In *NIPS*, pages 577–583, 2000.

[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.

[MR99] M. Müller and B. Rönz. Credit scoring using semiparametric methods. Sonderforschungsbereich 373 1999-93, Humboldt Universitaet, November 10, 1999 1999.

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

[MSW95] O. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

[Mus97] D. 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.

[NJ88] R. Nath and T. W. Jones. A variable selection criterion in the linear programming approaches to discriminant analysis. *Decision Sciences*, 19(3):554, 1988.

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

[PC01] F. Plastria and E. Carrizosa. Gauge distances and median hyperplanes. *Journal of Optimization Theory and Applications*, 110(1):173–182, 2001.

[Per04] S. Perron. *Applications jointes de l'optimisation combinatoire et globale.* PhD thesis, École Polytechnique de Montréal, 2004.

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

[Pir99] S. Piramuthu. Feature selection for financial credit-risk evaluation decisions. *Informs Journal on Computing*, 11(3):258–266, 1999.

[Pir04] S. Piramuthu. Evaluating feature selection methods for learning in data mining applications. *European Journal of Operational Research*, 156(2):483–494, 2004.

[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.

[RCW83] A. K. Reichert, Ch. Ch. Cho, and G. M. Wagner. An examination of the conceptual issues involved in developing credit-scoring models. *Journal of Business & Economic Statistics*, 1(2):101–114, 1983.

[RG94] E. Rosenberg and A. Gleit. Quantitative methods in credit management: a survey. *Operations Research*, 42(4):589–613, 1994.

[RS91] C. T. Ragsdale and A. Stam. Mathematical-programming formulations for the discriminant problem - an old dog does new tricks. *Decision Sciences*, 22(2):296–307, 1991.

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

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

[SG89] A. Steenackers and M. J. Goovaerts. A credit scoring model for personal loans. *Insurance Mathematics & Economics*, 8(1):31–34, 1989.

[SMH99] E. Stanghellini, K. J. McConway, and D. J. Hand. A discrete variable chain graph for applicants for credit. *Journal of the Royal Statistical Society Series C-Applied Statistics*, 48:239–251, 1999.

[Smi68] F. 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.

[SR92] A. Stam and C.T. Ragsdale. On the classification gap in mathematical-programming-based approaches to the discriminant problem. *Naval Research Logistics*, 39(4):545–559, 1992.

[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.

[Sta97] A. Stam. Nontraditional approaches to statistical classification: Some perspectives on l-p-norm methods. *Annals of Operations Research*, 74:1–36, 1997.

[TEC02] L. Thomas, D. B. Edelman, and J. Crook. *Credit scoring and its applications*. SIAM Monographs on Mathematical Modeling and Computation. SIAM, Philadelphia, PA, 2002.

[TH03] R. J. Till and D. J. Hand. Behavioural models of credit card usage. *Journal of Applied Statistics*, 30(10):1201–1220, 2003. DEC.

[Tho00] L. C. Thomas. A survey of credit and behavioral scoring: forecasting financial risk of lending to consumers. *International Journal of Forecasting*, 16:149–172, 2000.

[Tou71] G. Toussaint. Note on optimal selection of independent binary-valued features for pattern recognition. *IEEE Transactions on Information Theory*, IT-17:618, 1971.

[VVDP04] G. Verstraeten and D. Van Den Poel. The impact of sample bias on consumer credit scoring performance and profitability. Technical Report 04/232, Ghent University, 2004.

[Web02] A. Webb. *Statistical Pattern Recognition.* John Wiley & Sons, 2nd edition, 2002.

[WF05] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2nd edition, 2005.

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