Mathematical Programming in Data Mining [*]

O. L. Mangasarian[†]

Abstract

Mathematical programming approaches to three fundamental problems will be described: feature selection, clustering and robust representation. The feature selection problem considered is that of discriminating between two sets while recognizing irrelevant and redundant features and suppressing them. This creates a lean model that often generalizes better to new unseen data. Computational results on real data confirm improved generalization of leaner models. Clustering is exemplified by the unsupervised learning of patterns and clusters that may exist in a given database and is a useful tool for knowledge discovery in databases (KDD). A mathematical programming formulation of this problem is proposed that is theoretically justifiable and computationally implementable in a finite number of steps. A resulting k-Median Algorithm is utilized to discover very useful survival curves for breast cancer patients from a medical database. Robust representation is concerned with minimizing trained model degradation when applied to new problems. A novel approach is proposed that purposely tolerates a small error in the training process in order to avoid overfitting data that may contain errors. Examples of applications of these concepts are given.

1 Introduction

Mathematical programming, that is optimization subject to constraints, is a broad discipline that has been applied to a great variety of theoretical and applied problems such as operations research [29, 54], network problems [60, 53], game theory and economics [71, 35], engineering mechanics [57, 37] and more recently to machine learning [3, 61, 23, 48, 46]. In this paper we describe three recent mathematical-programming-based developments that are relevant to data mining: feature selection [45, 10], clustering [11] and robust representation [67]. We note at the outset that we do not plan to survey either the fields of data mining or mathematical programming, but rather highlight some recent and highly effective applications of the latter to the former. We will, however, point out other approaches that are mostly not based on mathematical programming.

The fundamental nonlinear programming problem [6, 44] consists of minimizing an objective function subject to inequality and equality constraints and is typically written as follows

$$\min_{x} \ f(x) \ \text{ subject to } \ g(x) \leq 0, \ h(x) = 0, \tag{1}$$

where x is an n-dimensional vector of real variables, f is a real-valued function of x, g and h are finite dimensional vector functions of x. If all the functions f, g and h are linear then the problem simplifies to a linear program [15, 52, 69] which is the classical problem of mathematical

[*]Mathematical Programming Technical Report 96-05, August 1996 – Revised November 1996 & March 1997. This material is based on research supported by National Science Foundation Grant CCR-9322479.

[†]Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706, email: *olvi@cs.wisc.edu*

programming. If x is two-dimensional, a linear program can be thought of as the problem of finding a lowest point (not necessarily unique) on a tilted plane surrounded by a piecewise-linear fence. Extremely efficient algorithms exist for the solution of linear programs. Thus reducing a problem to a single or finite sequence of linear programs is tantamount to solving the problem.

Another reason for emphasizing mathematical programming in this work is the very broad applicability of the optimization-under-constraints paradigm. A great variety of problems from many fields can be formulated and effectively solved as mathematical programs. According to the great eighteenth century mathematician Leonhard Euler: "Nothing happens in the universe that does not have a sense of either certain maximum or minimum" [68, p 1]. From the point of view of applicability to large-scale data mining problems, the proposed algorithms employ either linear programming (Sections 2 and 3) which is polynomial-time-solvable [36, 69], or convex quadratic programming (Section 4) which is also polynomial-time-solvable [69]. Extremely fast linear and quadratic programming codes [14] that are capable of solving linear programs with millions of variables [8, 40] and very large quadratic programs, make the proposed algorithms easily scalable and effective for solving a wide range of problems. One limitation however is that the problem features must be real numbers or easily mapped into real numbers. If some of the features are discrete and can be represented as integers, then the techniques of integer programming [24, 55, 14] can be employed. Integer programming approaches have been applied for example to clustering problems [64, 1], but will not be described here, principally because the combinatorial approach is fundamentally different than the analytical approach of optimization with real variables. Stochastic optimization methods based on simulated annealing have also been used in problems of inductive concept learning [50].

The problems considered in this paper are:

1. **Feature Selection** The feature selection problem treated is that of discriminating between two finite point sets in n-dimensional feature space by a separating plane that utilizes as few of the features as possible. The problem is formulated as a mathematical program with a parametric objective function and linear constraints [10]. A step function that appears in the objective function is approximated by a concave exponential on the nonnegative real line instead of the conventional sigmoid function of neural networks [28]. This leads to a very fast iterative linear-programming-based algorithm for solving the problem that terminates in a finite number of steps. On the Wisconsin Prognosis Breast Cancer (WPBC) [72, 51] database the proposed algorithm reduced cross-validation error on a cancer prognosis database by 35.4% while reducing problem features from 32 to 4.

2. **Clustering** The clustering problem considered in this paper is that of assigning m points in the n-dimensional real space R^n to k clusters. The problem is formulated as that of determining k centers in R^n such that the sum of distances of each point to the nearest center is minimized. Once the cluster centers are determined by a training set, a new point is assigned to the cluster with the nearest cluster center. If a polyhedral distance (such as the 1-norm distance) is used, the problem can be formulated as that of minimizing a piecewise-linear concave function on a polyhedral set which is shown to be equivalent to a bilinear program: minimizing the product of two linear functions on a set determined by satisfying a system of linear inequalities [11]. Although a bilinear program is a nonconvex optimization problem (i.e. minimizing a function that is *not* valley-like), a fast finite k-Median Algorithm consisting of solving few linear programs in closed form leads to a stationary point. Computational testing of this algorithm as a KDD tool [18] has been quite encouraging. On the Wisconsin Prognosis Breast Cancer Database (WPBC), distinct and clinically important

survival curves were discovered from the database by the k-Median Algorithm, whereas the traditional k-Mean Algorithm [32, 64], which uses the *square* of the 2-norm distance, thus emphasizing outliers, failed to obtain such distinct survival curves for the same database. On four other publicly available databases each of the k-Median and k-Mean Algorithms did best on two of the databases.

3. **Robust Representation** This problem deals with modeling a system of relations within a database in a manner that preserves, to the extent possible, the validity of the representation when the data on which the model is based changes. This problem is closely related to the generalization problem of machine learning of how to train a system on a given training set so as to improve generalization on a new unseen testing set [39, 63, 73]. We use here a simple linear model [67] and will show that if a sufficiently small error τ is *purposely* tolerated in constructing the model, then for a broad class of perturbations the model will be a more accurate representation than one obtained by a conventional zero error tolerance. A simple example demonstrates this result.

1.1 Notation

We summarize below notation and background material used in this paper.

- All vectors will be column vectors unless transposed to a row vector by a superscript T.
- For a vector x in the n-dimensional real space R^n, $|x|$ will denote a vector of absolute values of components $x_i,\ i = 1, \ldots, n$ of x.
- The base of the natural logarithm will be denoted by ε, and for $y \in R^m$, ε^{-y} will denote a vector in R^m with component $\varepsilon^{-y_i},\ i = 1, \ldots, m$.
- For $x \in R^n$ and $1 \leq p < \infty$, the norm $\|x\|_p$ will denote the p-norm, that is $(\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$.
- The notation $A \in R^{m\times n}$ will signify a real $m\times n$ matrix. For such a matrix, A^T will denote the transpose, A_i will denote row i and A_I will denote those rows A_i such that $i \in I \subset \{1, \ldots, m\}$ for a given subset I of $\{1, \ldots, m\}$.
- A vector of ones in a real space of arbitrary dimension will be denoted by e. A vector of zeros in a real space of arbitrary dimension will be denoted by 0.
- If x and y are vectors in R^n, the notation $\min\{x, y\}$ will denote the vector in R^n of componentwise minimum of x_i and y_i, $i = 1, \ldots, n$.
- The notation $\arg\min\limits_{x\in S} f(x)$ will denote the set of minimizers of $f(x)$ on the set S. Similarly $\arg\operatorname{vertex}\min\limits_{x\in S} f(x)$ will denote the set of vertex minimizers of $f(x)$ on a polyhedral set S.
- A polyhedral set S in R^n is the the intersection of a finite number of closed halfspaces in R^n.
- A vertex of a polyhedral set S is a boundary point of S that lies on the intersection of n linearly independent planes constituting the boundary. Typically $S = \{x \mid Ax \geq b\}$, where $A \in R^{m\times n}$, $b \in R^m$ and a point $p \in S$ is a vertex of S if for some subset I of $\{1, \ldots, m\}$, $A_I p = b_I$ and $A_I \in R^{n\times n}$ is nonsingular.

- A separating plane, with respect to two given point sets $\mathcal{A}$ and $\mathcal{B}$ in R^n, is a plane that attempts to separate R^n into two halfspaces such that each open halfspace contains points mostly of $\mathcal{A}$ or $\mathcal{B}$.
- Alternatively, a separating plane can be interpreted as a classical perceptron [62, 43] with a threshold determined by the distance of the plane to the origin, and the incoming arc weights of the perceptron determined by the components of the normal vector to the plane.

2 Feature Selection

Feature selection (or extraction) attempts to use the simplest model to describe the essence of a phenomenon. Hence it can be considered as an application of Occam's "law of parsimony", also known as Occam's Razor [65, 9], which states: "What can be done with fewer [assumptions] is done in vain with more". There are statistical [22], machine learning [39, 33] as well as mathematical programming [12, 45, 10] approaches to the feature selection problem. In this work we shall deal principally with the latter because of the novelty of the approach and it effectiveness.

The problem that we shall address is the *binary classification problem*, i.e. the problem of discriminating between two given point sets $\mathcal{A}$ and $\mathcal{B}$ in the n-dimensional real space R^n by using as few of the n dimensions of the space as possible. For example in the medical application described at the end of this section, we attempt to discriminate, with as few of the features as possible, between breast cancer patients that had a recurrence of the disease within two years of diagnosis and those who had not. The model that we shall adopt will be that of a perceptron or linear threshold unit (LTU) that employs as few of the features of the given problem as possible. Extensions to more complex models leading to compact and accurate decision trees have also been made [12]. Geometrically our approach corresponds to constructing a plane in R^n defined by

$$P := \{x \mid x \in R^n,\ x^T w = \gamma\}, \tag{2}$$

with normal $w \in R^n$ and distance $\frac{|\gamma|}{\|w\|_2}$ to the origin, while suppressing as many of the components of w as possible. In addition, the set $\mathcal{A}$ must lie, to the extent possible, in the open halfspace

$$\{x \mid x \in R^n,\ x^T w > \gamma\}, \tag{3}$$

and the set $\mathcal{B}$ in the open halfspace

$$\{x \mid x \in R^n,\ x^T w < \gamma\}. \tag{4}$$

This corresponds to an LTU with a threshold γ and and incoming arc weights $w \in R^n$, with as many of the weights as possible set to zero. If we represent the set $\mathcal{A}$ by the matrix $A \in R^{m\times n}$ and the set $\mathcal{B}$ by the matrix $B \in R^{k\times n}$, then the problem is to find $\gamma \in R$ and $w \in R^n$, with as many components equal to zero as possible, such that the following inequalities are satisfied in some best sense:

$$Aw > e\gamma,\ Bw < e\gamma, \tag{5}$$

where e is a vector of ones. Because linear programming cannot handle strict inequality constraints we rescale (5) as follows. We divide the variables (w, γ) by the positive quantity

$$\min_{i=1,\ldots,m,j=1,\ldots,k} \{A_i w - \gamma,\ -B_j w + \gamma\},$$

and call the rescaled variables (with notational economy in mind and a slight abuse of notation) (w, γ) again, then (5) is equivalent to:

$$Aw \geq e\gamma + e, \; Bw \leq e\gamma - e. \tag{6}$$

Since these inequalities may not have a solution in general, one resorts to satisfying them in some best approximate sense by minimizing an average sum of their violations. This leads to the following Robust Linear Programming formulation [4]:

$$\textbf{RLP} \quad \min_{w,\gamma,y,z} \left\{ \frac{e^T y}{m} + \frac{e^T z}{k} \;\middle|\; \begin{array}{c} -Aw + e\gamma + e \leq y, \\ Bw - e\gamma + e \leq z, \\ y \geq 0, z \geq 0 \end{array} \right\}. \tag{7}$$

Robustness here refers to the fact that the useless null vector $(w = 0)$ is naturally excluded as a solution of (7), which is not the case in other linear programming formulations of this problem [41, 66, 26, 25]. Note that because of the constraints of the problem, the variables y and z will satisfy the following conditions:

$$y \geq \min\{0, \; -Aw + e\gamma + e\} \text{ and } z \geq \min\{0, \; Bw - e\gamma + e\}.$$

Hence minimizing $\frac{e^T}{m} + \frac{e^T z}{k}$ will force the satisfaction in some best sense of (6), and equivalently (5), by minimizing the average violations of (6):

$$\frac{1}{m} \| \min\{0, \; -Aw + e\gamma + e\} \|_1 + \frac{1}{k} \| \min\{0, Bw - e\gamma + e\} \|_1 \geq 0,$$

which will be zero if and only if (6), or equivalently (5), is exactly satisfied. The linear programming formulation (7) which has a number of natural theoretical properties including robustness is also very effective computationally [4, 49]. However it does *not* address the problem of suppressing irrelevant features. In order suppress such features, the objective function of (7), which merely measures the average sum of the violations of the inequalities (6), is modified so as to also suppress as many of the components of the weight vector w as possible. This is achieved by weighting the original objective of (7) by $1 - \lambda$ with $\lambda \in (0, 1)$, and weighting by λ an exponential function approximation of the absolute value v of the weight vector w. This exponential function, $e^T(e - \varepsilon^{-\alpha v})$, approximates the 1-norm of a step function of v and leads to the following mathematical program with a concave objective function and linear constraints:(<u>F</u>eature <u>S</u>election Conca<u>v</u>e)

$$(\mathbf{FSV}) \min_{w,\gamma,y,z,v} \left\{ (1 - \lambda)\left(\frac{e^T y}{m} + \frac{e^T z}{k}\right) + \lambda e^T (e - \varepsilon^{-\alpha v}) \;\middle|\; \begin{array}{c} -Aw + e\gamma + e \leq y, \\ Bw - e\gamma + e \leq z, \\ y \geq 0, z \geq 0, \\ -v \leq w \leq v \end{array} \right\}, \; \lambda \in [0, 1) \tag{8}$$

When $\lambda = 0$ the problem degenerates to the robust (that is $w \neq 0$) linear program of (7) which obtains a plane P (2) that separates the sets $\mathcal{A}$ and $\mathcal{B}$ in an optimal fashion without regard to feature suppression. When $\lambda > 0$, then in addition to the objective of separating $\mathcal{A}$ and $\mathcal{B}$, we attempt to suppress as many of the components of w as possible by minimizing an exponential smoothing of the step function on the nonnegative real line for each v_i: $(1 - \varepsilon^{-\alpha v_i})$, $i = 1, \ldots, n$, where ε is the base of the natural logarithms, and v is the absolute value of w. In most our applications a value of $\alpha = 5$ was sufficient to make the exponential a good approximation of the step function to force suppression of unnecessary components of w. As described below in Algorithm 2.1 the parameter λ is chosen to give the best cross-validated error. For small values of λ it can be shown theoretically [47] that the minimization problem (8) picks that solution of the Robust Linear Program (7) that minimizes the exponential term of (8), and hence solves the RLP (7) while suppressing redundant components of w. Because the objective function of (8) is a concave function bounded below by zero on the nonempty polyhedral set of (8), it follows that it has a vertex solution if the feasible region does not contain lines extending to infinity in both directions [59, Corollaries 32.3.3, 32.3.4]. (Excluding such lines can be readily accomplished by a simple transformation of the variables (w, γ) into the nonnegative variables (w^1, γ^1, ζ^1) using the standard transformation $w = w^1 - e\zeta^1$, $\gamma = \gamma^1 - \zeta^1$. For the sake of simplicity and because it is not needed computationally, we shall forgo this transformation here.) A fast, finitely-terminating successive linear programming algorithm has been proposed for solving this problem [10] as follows.

Algorithm 2.1 Successive Linearization Algorithm (SLA) for FSV (8). *Choose $\lambda \in [0, 1)$. Start with a random $(w^0, \gamma^0, y^0, z^0, v^0)$. Having $(w^i, \gamma^i, y^i, z^i, v^i)$ determine the next iterate by solving the linear program:*

$$(w^{i+1}, \gamma^{i+1}, y^{i+1}, z^{i+1}, v^{i+1}) \in \arg vertex \min_{w,\gamma,y,z,v} \left\{ \begin{array}{c|c} (1-\lambda)(\frac{e^Ty}{m} + \frac{e^Tz}{k}) & \begin{array}{c} -Aw + e\gamma + e \leq y, \\ Bw - e\gamma + e \leq z, \end{array} \\ +\lambda\alpha\varepsilon^{-\alpha v^i}(v - v^i) & \begin{array}{c} y \geq 0,\ z \geq 0, \\ -v \leq w \leq v \end{array} \end{array} \right\} \tag{9}$$

Stop when

$$(1-\lambda)(\frac{e^T(y^{i+1} - y^i)}{m} + \frac{e^T(z^{i+1} - z^i)}{k}) + \lambda\alpha\varepsilon^{-\alpha v^i}(v^{i+1} - v^i) = 0. \tag{10}$$

Comment: The parameter α was set to 5. The parameter λ was chosen in the set {0, 0.05, 0.10, ..., 1.00}, with the desired λ being the one achieving the best cross-validated separation.

It has been shown [45, Theorem 4.2] that this algorithm terminates in a finite number of steps, typically five or six, at a global solution or a stationary point satisfying a necessary optimality condition.

This algorithm was tested on the 32-feature Wisconsin Prognostic Breast Cancer (WPBC) database [72, 51] which was collected from 28 patients for which cancer recurred within two years, and 119 patients for which cancer did not recur within two years. Thus in the terminology of our formulation (8), $n = 32$, $m = 28$ and $k = 118$. For this problem the separating plane obtained by the Successive Linearization Algorithm 2.1 with $\lambda = 0.05$ used only 4 features out of 32, while increasing tenfold cross-validation correctness by 35.4% [10]. If other values of the parameter λ are used in the Successive Linearization Algorithm 2.1, then the number of features that determine the separating plane will vary between 1 and 32. The effect of using a different number of features is shown in Figure 1 which shows a plot of tenfold cross-validation correctness corresponding to the number of features used. As just indicated, Figure 1 shows that the best tenfold correctness occurs at four features.

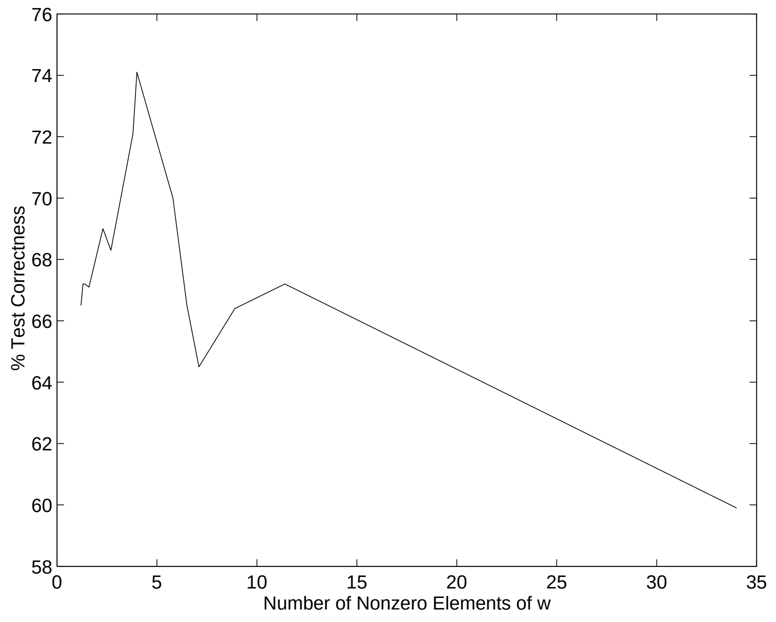


Figure 1: Feature Selection in the Prognosis Problem: Tenfold cross-validation correctness versus number of features selected by the Successive Linearization Algorithm 2.1

3 Clustering via Mathematical Programming

The unsupervised assignment of elements of a given set into groups or clusters of like points, is the objective of cluster analysis. There are many approaches to this problem, including statistical [32], machine learning [20], integer and mathematical programming approaches [64, 1, 58, 11]. We shall describe here the recent approach of [11] that utilizes a fast bilinear programming approach: minimizing the product of two linear functions on a set defined by linear inequalities. A principal motivation behind our mathematical programming approach is a precise and concise statement of the clustering problem as a concave minimization problem (11) which has not been given before. We note that a concave minimization problem which involves minimizing a concave function (mountain-like function) on a polyhedral set S is more difficult than minimizing a convex function (valley-like function) on S because the former may have many local minima at vertices of S that are not global minima, whereas for the latter each local minimum is a global minimum. Nevertheless for our clustering application the concave formulation is very effective in discovering well separated survival curves by determining k cluster centers such that the sum of the 1-norm distances of each point in a given database to the nearest cluster center is minimized. A new point is then assigned to the cluster with center nearest to the point. This simple formulation can be restated as a bilinear program (12) that leads to a fast k-Median Algorithm 3.1. A reformulation of (11) using the square of the 2-norm instead of the 1-norm, leads to the k-Mean Algorithm [32, 64]. Although it is not the intention here to carry out a detailed comparative study of these two clustering algorithms, the k-Median Algorithm, as described below, does give well separated survival curves for breast cancer patients whereas the k-Mean algorithm does not. On four other publicly available databases, each of the k-Median and k-Mean Algorithms did best on two of the databases. This indicates the potential of the k-Median Algorithm as a KDD tool. We describe the mathematical programming approach now.

For a given set $\mathcal{A}$ of m points in R^n represented by the matrix $A \in R^{m\times n}$ and a number k of desired clusters, we formulate the clustering problem as follows. Find cluster centers C_ℓ, $\ell = 1,\ldots,k$, in R^n such that the sum of the minima over $\ell \in \{1,\ldots,k\}$ of the 1-norm distance between each point A_i, $i = 1,\ldots,m$, and the cluster centers C_ℓ, $\ell = 1,\ldots,k$, is minimized. More specifically we need to solve the following mathematical program:

$$\begin{array}{ll} \underset{C,D}{\text{minimize}} & \displaystyle\sum_{i=1}^{m} \min_{\ell=1,\ldots,k} \{e^T D_{i\ell}\} \\ \text{subject to} & -D_{i\ell} \le A_i^T - C_\ell \le D_{i\ell},\ i=1,\ldots,m,\ \ell=1,\ldots k \end{array} \tag{11}$$

Here $D_{i\ell} \in R^n$, is a dummy variable that bounds the components of the difference $A_i^T - C_\ell$ between point A_i^T and center C_ℓ, and e is an $n \times 1$ vector of ones in R^n. Hence $e^T D_{i\ell}$ bounds the 1-norm distance between A_i and C_ℓ. We note that just as in the case of robust regression [31],[27, pp 82-87], the use of the 1-norm here to measure the error criterion leads to insensitivity to outliers such as those resulting from distributions with pronounced tails. We also note that since the objective function of (11) is the minimum of k linear (and hence concave) functions, it is a piecewise-linear concave function [44, Corollary 4.1.14]. This is not the case for the 2-norm or p-norm, $p \neq 1$. Although (11) is NP-hard, it can be reformulated as the following bilinear program which can be solved effectively by using a k-Median Algorithm that consists of solving a succession of simple linear programs in closed form. We state the bilinear programming formulation and k-Median Algorithm for solving the clustering problem.

Proposition 3.1 Clustering as a Bilinear Program *The clustering problem (11) is equivalent to the following bilinear program:*

$$\begin{array}{lll} \underset{C_\ell\in R^n, D_{i\ell}\in R^n, T_{i\ell}\in R^n}{\text{minimize}} & \multicolumn{2}{l}{\sum_{i=1}^{m}\sum_{\ell=1}^{k} e^T D_{i\ell} T_{i\ell}} \\ \text{subject to} & \multicolumn{2}{l}{-D_{i\ell} \le A_i^T - C_\ell \le D_{i\ell}, i=1\ldots,m,\ \ell=1,\ldots,k} \\ & \sum_{\ell=1}^{k} T_{i\ell} = 1 & T_{i\ell} \ge 0,\ i=1,\ldots,m,\ \ell=1,\ldots,k \end{array} \tag{12}$$

This essentially obvious result [11, Proposition 2.2] can be seen from the fact that, for a fixed i, setting all the components of $T_{i\ell}$, $\ell = 1,\ldots,k$ equal to zero except one corresponding to a smallest $e^T D_{i\ell}$, with respect to ℓ, equal to 1, leads to the objective function of (11) from that of (12). Note that the constraints of (12) are uncoupled in the variables (C, D) and the variable T. Hence the Uncoupled Bilinear Program Algorithm UBPA [5, Algorithm 2.1] is applicable. Simply stated, this algorithm alternates between solving a linear program in the variable T and a linear program in the variables (C, D). The algorithm terminates in a finite number of iterations at a stationary point satisfying the minimum principle necessary optimality condition for problem (12) [5, Theorem 2.1]. We note however, because of the simple structure the bilinear program (12), the two linear programs can be solved explicitly in closed form. This leads to the following algorithmic implementation.

Algorithm 3.1 k-Median Algorithm *Given the cluster centers $C_1^j,\ldots,C_k^j$ at iteration j, compute $C_1^{j+1},\ldots,C_k^{j+1}$ by the following two steps:*

(a) Cluster Assignment: *For each A_i^T, $i = 1,\ldots m$, determine $\ell(i)$ such that $C_{\ell(i)}^j$ is closest to A_i^T in the one norm.*

(b) Cluster Center Update: *For $\ell = 1,\ldots,k$ choose C_ℓ^{j+1} as a median of all A_i^T assigned to C_ℓ^j.*

Stop when $C_\ell^{j+1} = C_\ell^j$. Assign each point to a cluster whose center is closest in the 1-*norm to the point.*

Although the k-Median Algorithm is similar to the k-Mean Algorithm wherein the 2-norm distance is used [64, 22], it differs from it computationally, and theoretically. In fact, the underlying problem (12) of the k-Median Algorithm is a concave minimization on a polyhedral set while the corresponding problem for a two- or p-norm, $p \neq 1$, is:

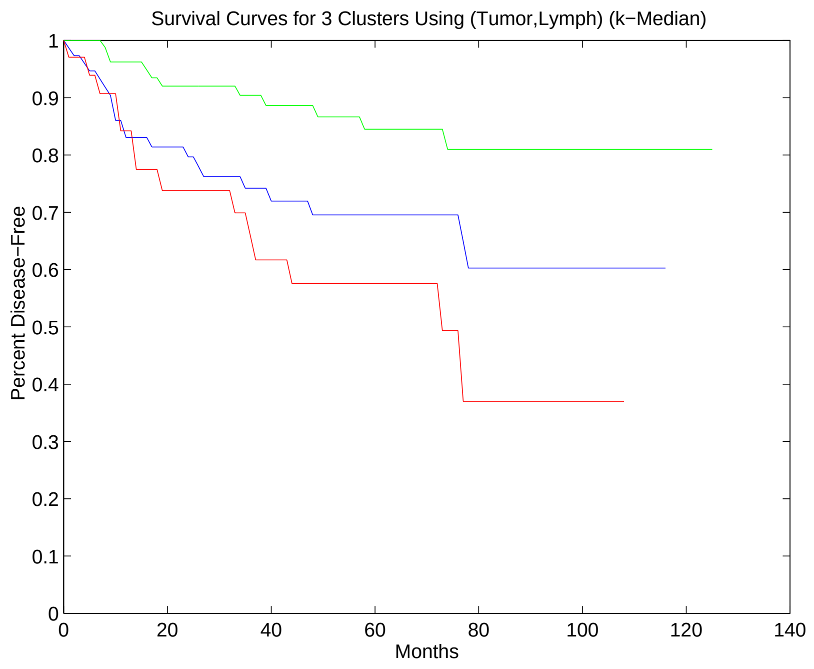
$$\begin{array}{ll} \underset{C,D}{\text{minimize}} & \sum_{i=1}^{m} \min_{\ell=1,\dots,k} \|D_{i\ell}\|_p \\ \text{subject to} & -D_{i\ell} \leq A_i^T - C_\ell \leq D_{i\ell}, i = 1\dots, m,\ \ell = 1, \dots, k. \end{array} \tag{13}$$

This is not a concave minimization on a polyhedral set, because the minimum of a set of convex functions is not in general concave. We also note that the k-Mean Algorithm finds a stationary point *not* of problem (13) with $p = 2$, but of the same problem except that $\|D_{i\ell}\|_2$ is replaced by $\|D_{i\ell}\|_2^2$ and thus favoring outliers. Without this squared distance term, the subproblem of the k-Mean Algorithm becomes the considerably harder Weber problem [56, 13] which locates a center in R^n closest in sum of Euclidean distances (not their squares!) to a finite set of given points. The Weber problem has no closed form solution. However, using the mean as a cluster center of points assigned to the cluster, as done in the k-Mean Algorithm, minimizes the sum of the *squares* of the distances from the cluster center to the points.

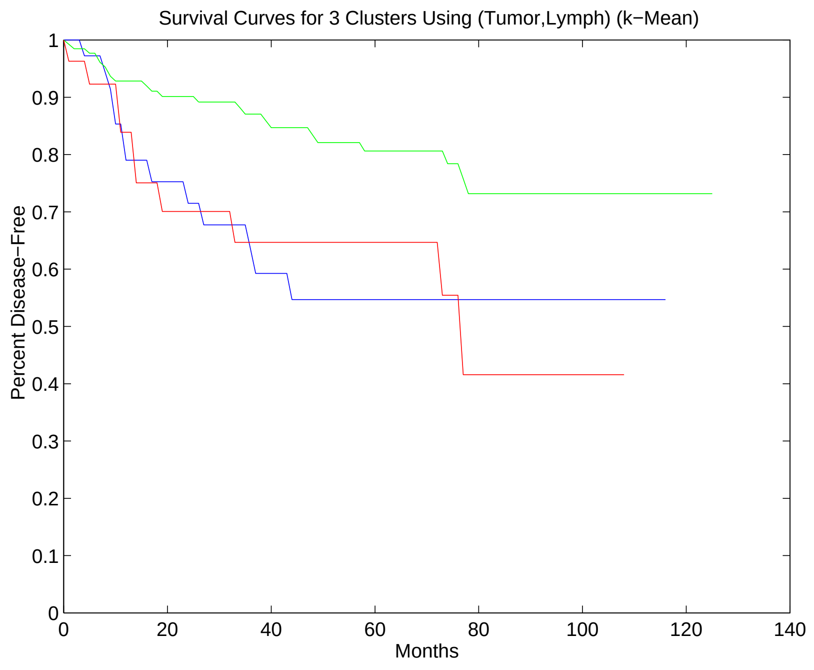
Because there is no guaranteed way to ensure global optimality of the solution obtained by either the k-Median or k-Mean Algorithms, different starting points can be used to initiate the algorithm. Random starting cluster centers or some other heuristic can be used such as placing k initial centers along the coordinate axes at densest, second densest, ..., k densest intervals on the axes. The latter heuristic was used in our computational results.

To test the effectiveness of the k-Median Algorithm it was used as a KDD tool [18] to mine the Wisconsin Prognostic Breast Cancer Database (WPBC) in order to discover medical knowledge. For such medical databases, extracting well-separated survival curves provides an essential prognostic tool. Survival curves [34, 38] give expected percent of surviving patients as a function of time. The k-Median Algorithm was applied to WPBC to extract such curves. Survival curves were constructed for 194 patients using two clinically available features for each patient: tumor size and number of cancerous lymph nodes excised. Using $k = 3$, the k-Median Algorithm separated the points into 3 clusters. The survival curve for each cluster is depicted in Figure 2(a). The key observation to make here is that curves are well separated, and hence the clusters can be used as prognostic indicators to assign a survival curve to a patient depending on the cluster into which the patient falls. By contrast, the k-Mean Algorithm obtained poorly separated survival curves as shown in Figure 2(b), and hence are not useful for prognosis.

Another comparison of the k-Median and k-Mean Algorithms was performed on databases with known classes. Correctness was measured by the ratio of the sum of the number of majority points in each cluster to the total number of points m in the data set. Table 1 shows results averaged over ten random starts for four databases from the Irvine repository of databases [51]. We note that for two of the databases the k-Median gave better correctness than the k-Mean and for the other two the k-Mean was better.



(a) k-Median



(b) k-Mean

Figure 2: Survival curves for the 3 clusters of 194 cancer patients obtained by the k-Median and k-Mean Algorithms

Algorithm ↓ Database →	WDBC	Cleveland	Votes	Star/Galaxy-Bright
Unsupervised k-Median	93.2%	80.6%	84.6%	87.6%
Unsupervised k-Mean	91.1%	83.1%	85.5%	85.6%

Table 1 Training set correctness using the unsupervised k-Median and k-Mean Algorithms and the supervised Robust LP on four databases

4 Robust Representation

We consider now the problem of how to generate a robust representation of a system so that the model remains valid under a class of data perturbation. This problem is closely related to the generalization problem of machine learning of how to train a system on a given training set so as to improve generalization on a new unseen testing set [39, 63, 73]. We shall concentrate on some recent results [67] obtained for a simple linear model and which make essential use of mathematical programming ideas. These ideas, although rigorously established for a simple linear model here, are likely to extend to more complex systems. In a somewhat related approach Vapnik has proposed a quadratic program for constructing a plane to obtain a smallest probability of error for separating two point sets [70, Section 5.4]. Bennett and Bredensteiner [2] have formulated a similar problem using linear programming. Vapnik [70, Section 5.9] also makes use of Huber's robust regression ideas [30] and extends the latter's robust regression loss function [70, p 152] by adding an ϵ-insensitive zone to it. This ϵ-insensitive zone is similar to our τ-tolerance zone (see (17) and (18) below) wherein errors are disregarded if they fall within the band $[-\tau, \tau]$. Another similarity with Vapnik's work is the presence of a regularization term $\frac{\epsilon}{2}\|x\|_2^2$ in our minimization (17) problem (introduced here in order to make the solution unique and to rigorously derive our generalization theorems) and Vapnik's bounding his weight vector in order to control the VC dimension [70, p 128, Theorem 5.1]. However, what our approach provides here that is novel, are precise deterministic conditions (Propositions 4.1 and 4.2 below) under which tolerating a τ-insensitivity zone will give better generalization results than the conventional zero-tolerance that is used in an ordinary least squares approach.

The model that we shall consider here consists of the *training set* $\{A, a\}$ where A is a given $m \times n$ real matrix and a is a given $m \times 1$ real vector. A vector x in R^n is to be "learned" such that the linear system

$$Ax = a, \tag{14}$$

which does not have an exact solution, is satisfied in some approximate fashion, and such that the error in satisfying

$$Cx = c, \tag{15}$$

for some unseen *testing set* $(C, c) \in R^{k\times n} \times R^k$, is minimized. Of course, if we disregard the testing set error (15), the problem becomes the standard least-norm problem:

$$\min_{x\in R^n} \|Ax - a\|, \tag{16}$$

where $\|\cdot\|$ is some norm on R^m. However with an eye to possible perturbations in the given training set $\{A, a\}$, we pose the following motivational question: If the vector a of the training set is known only to an accuracy of τ, where τ is some small positive number, does it make sense to attempt to drive the error to zero as is done in (16), or is it not better to tolerate errors in the satisfaction of $Ax = a$ up to a magnitude of τ? In other words, instead of (14), we should try to satisfy the following system of inequalities, in some best sense:

$$-e\tau \leq Ax - a \leq e\tau \tag{17}$$

To do that, we solve the following regularized quadratic program for some nonnegative τ and a small positive ϵ:

$$\begin{array}{ll} \underset{x,y,z}{\text{minimize}} & \frac{1}{2}\|y\|_2^2 + \frac{1}{2}\|z\|_2^2 + \frac{\epsilon}{2}\|x\|_2^2 \\ \text{subject to} & \begin{array}{c} -z - e\tau \leq Ax - a \leq e\tau + y \\ y, z \geq 0. \end{array} \end{array} \tag{18}$$

Here y and z are the errors in satisfying the inequalities of (17) and ϵ is a small fixed positive regularization constant that ensures the uniqueness of the x component of the solution. Although ϵ was held fixed in our computational experiments, it is possible to optimize its value by cross-validation on a tuning set in order to obtain better generalization. We note immediately, that if $\tau = 0$, problem (18) degenerates to the regularized classical least squares problem:

$$\min_{x \in R^n} \frac{1}{2}\|Ax - a\|_2^2 + \frac{\epsilon}{2}\|x\|_2^2. \tag{19}$$

The key question to ask here, is this: Under what conditions does a solution $x(\tau)$ of (18), for some $\tau > 0$ give a smaller error than $x(0)$ on a testing set? We are able to give an answer to this question and corroborate it computationally [67], by considering a general testing set $(C, c) \in R^{k \times n} \times R^k$ for the problem (15) as well as a simpler testing set, where only the right side of (14) is perturbed. We first restrict ourselves to the latter and simpler perturbation, that is:

$$Ax = a + p, \tag{20}$$

where p is some arbitrary fixed perturbation in R^m, and consider the following associated error function:

$$f(\tau) := \frac{1}{2}\|Ax(\tau) - a - p\|_2^2. \tag{21}$$

In particular we would like to know when is $f(0)$ *not* a local minimum of $f(\tau)$ on the set $\{\tau \mid \tau \geq 0\}$. In fact we are only interested in the τ-interval $[0, \hat{\tau}]$, where $\hat{\tau}$ is defined by

$$\hat{\tau} := \min_x \|Ax - a\|_\infty, \tag{22}$$

because the minimum value of (18) approaches zero, for $\tau \geq \hat{\tau}$, as ϵ approaches zero. The following proposition gives a sufficient condition which ensures that solving (18), for some positive τ, produces an $x(\tau)$ that generalizes better on the system (20) than that obtained by solving a plain regularized least squares problem (19), that is $f(\bar{\tau}) < f(0)$ for some $\bar{\tau} > 0$.

Proposition 4.1 Robust Representation of $Ax = a + p$ via (18) *[67]. For a solution $x(\tau)$ of (18) the testing set error function $f(\tau)$ of (21) has a strict local maximum at 0 and a global minimum on $[0, \hat{\tau}]$, where $\hat{\tau}$ is defined by (22), at some $\bar{\tau} > 0$, whenever*

$$(\epsilon x(0) + A^T p)^T (x(\tau) - x(0)) > 0 \tag{23}$$

for some $\tau \in (0, \tilde{\tau}]$, for some sufficiently small $\tilde{\tau}$.

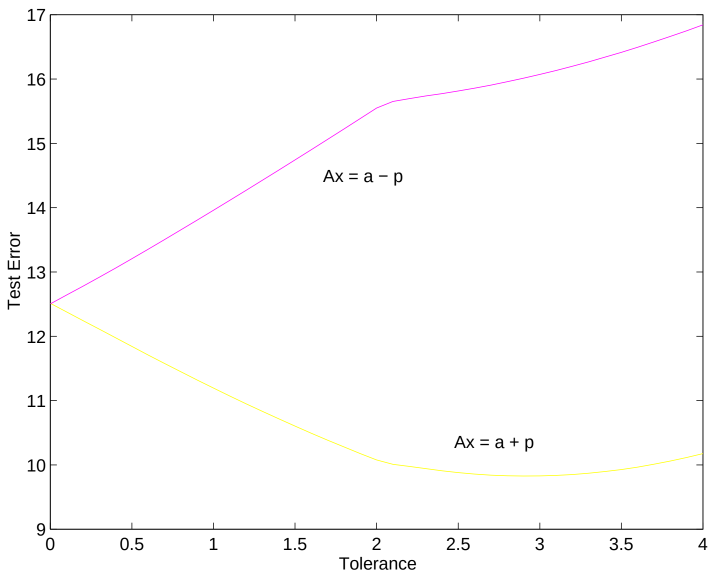


Figure 3: A computational example of Proposition 4.1. The bottom curve depicts a decreasing test error $f(\tau)$ of (21) for the perturbed system $Ax = a + p$ satisfying condition (23). The top curve demonstrates an increasing error effect of violating (23) for sufficiently small ϵ.

Computational results carried out in [67] have corroborated the improved generalization results of Proposition 4.1 above. We depict in Figure 3 a simple numerical example that uses the training model (14), where the vector x is learned with various error tolerances τ (18) and is then tested on the perturbed system $Ax = a + p$. In the learning situation depicted by the bottom curve of Figure 3, $(\epsilon x(0) + A^T p)$ makes an acute angle with $(x(\tau) - x(0))$ in the neighborhood of $\tau = 0$. Hence, as τ increases, the test error in satisfying $Ax = a + p$ decreases. In the upper curve, we show a perturbation for which the angle is reversed: the testing model is $Ax = a - p$, using the same perturbation p. Here the testing error goes up as τ increases away from zero.

We conclude this section by extending Proposition 4.1 to a more general testing model

$$Cx = c \tag{24}$$

where $C \in R^{k\times n}$ and $c \in R^k$ are chosen arbitrarily. To do this we define a corresponding error function $g(\tau)$ to (21) which measures the error in satisfying (24) by the $x(\tau)$ learned by solving the regularized quadratic program (18). We thus have the error

$$g(\tau) := \frac{1}{2}\|Cx(\tau) - c\|_2^2 \tag{25}$$

For this very general formulation we are able to give the following result that tells us when tolerant training does lead to improved generalization.

Proposition 4.2 Improved generalization with positive tolerance for testing model $Cx = c$ *[67]. Let $x(\tau)$ be defined by the tolerant training of $Ax = a$ by the regularized quadratic program (18) with tolerance $\tau \geq 0$. Let $g(\tau)$ denote the error generated by $x(\tau)$ in the testing model*

$Cx = c$, defined by (25). The zero-tolerance error $g(0)$ is a local maximum of $g(\tau)$ over the set $\{\tau : \tau \geq 0\}$ whenever

$$\|r(0)\|_2^2 > r(\tau)^T r(0) \text{ for some } \tau \in (0, \tilde{\tau}], \text{ for some } \tilde{\tau} > 0, \tag{26}$$

where $r(\tau)$ is the residual vector defined by

$$r(\tau) := Cx(\tau) - c. \tag{27}$$

Furthermore, the testing set error function $g(\tau)$ of (25) has a global minimum on $[0, \hat{\tau}]$, for $\hat{\tau}$ defined by (22), at some $\bar{\tau} > 0$, whenever condition (26) holds and $\hat{\tau} > 0$.

5 Conclusion

A number of ideas based on mathematical programming have been proposed for the solution of the fundamental problems of feature selection, clustering and robust representation. Examples of applications of these ideas have been given to show their effectiveness. We discuss now some issues associated with these approaches.

All the methods here use real variables. Even though the class of problems falling in this category is quite broad, this requirement imposes a restriction on the type of problems that can be handled. Nevertheless the proposed methods can be applied to problems with discrete variables if one is willing to use the techniques of integer and mixed integer programming [55, 21] which are more difficult. In fact one of the proposed algorithms, the k-Median Algorithm, whose finite termination is established for problems with real variables, is directly applicable with no change to problems with ordered discrete variables such as integers. How well it performs on such problems would be an interesting problem to examine.

Another important practical issue is scalability. As mentioned earlier, since linear programs with millions of variables can be solved with present day state-of-the-art methods, large-scale databases are amenable to the proposed linear-programming-based methods. In addition there is a large body of literature on the parallel solution and decomposition of large scale mathematical programs [7, 16, 17, 19] where for many of the algorithms only part of the data is loaded into memory at a time. Such parallel and decomposition algorithms extend further the applicability of the proposed methods to very large scale databases.

From a numerical standpoint, mathematical programming codes, and especially linear and quadratic programming codes, are reliable and robust codes that have been in a constant state of improvement over last fifty years. The well-understood polynomial-time finite termination of linear and quadratic programming interior methods has led to very powerful and reliable commercial software such as CPLEX [14] that can easily and reliably implement all the proposed algorithms.

Finally we point out that although a linear model was used for both the feature selection and robust representation models, nonlinear models that are linear in their parameters, e.g. quadratic surfaces, can be easily transformed into a linear system, as was done for example in [41] where quadratic separation was achieved by linear programming. However in some inherently nonlinear problems where for example the parameters of a separating surface appear nonlinearly, one may have to resort to nonlinear models and the theory and algorithms of nonlinear programming [42, 6]. This would again be a promising problem to pursue.

We conclude with the hope that the problems solved demonstrate the theoretical and computational potential of mathematical programming as a versatile and effective tool for solving important problems in data mining and knowledge discovery in databases.

References

[1] K. Al-Sultan. A Tabu search approach to the clustering problem. *Pattern Recognition*, 28(9):1443–1451, 1995.

[2] K. P. Bennett and E. J. Bredensteiner. Geometry in learning. In C. Gorini, E. Hart, W. Meyer, and T. Phillips, editors, *Geometry at Work*, Washington, D.C., 1997. Mathematical Association of America. To appear, www.rpi.math.edu/ bennek/geometry2.ps.

[3] K. P. Bennett and O. L. Mangasarian. Neural network training via linear programming. In P. M. Pardalos, editor, *Advances in Optimization and Parallel Computing*, pages 56–67, Amsterdam, 1992. North Holland.

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

[5] K. P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. *Computational Optimization & Applications*, 2:207–227, 1993.

[6] D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, 1995.

[7] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation.* Prentice–Hall, Inc, Englewood Cliffs, New Jersey, 1989.

[8] R. E. Bixby, J. W. Gregory, I. J. Lustig, R. E. Marsten, and D. F. Shanno. Very large-scale linear programming: a case study combining interior point and simplex methods. *Operations Research*, 40:885–897, 1992.

[9] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[10] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Feature selection via mathematical programming. Technical Report 95-21, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, December 1995. *INFORMS Journal on Computing*, submitted. Available by ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-21.ps.Z.

[11] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. Technical Report 96-03, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, May 1996. Advances in Neural Information Processing Systems 9, MIT Press, Cambridge, MA 1997, to appear. Available by ftp://ftp.cs.wisc.edu/math-prog/tech-reports/96-03.ps.Z.

[12] E. J. Bredensteiner and K. P. Bennett. Feature minimization within decision trees. Department of Mathematical Sciences Math Report No. 218, Rensselaer Polytechnic Institute, Troy, NY 12180, 1995.

[13] F. Cordellier and J. Ch. Fiorot. On the Fermat-Weber problem with convex cost functionals. *Mathematical Programming*, 14:295–311, 1978.

[14] CPLEX Optimization Inc., Incline Village, Nevada. *Using the CPLEX(TM) Linear Optimizer and CPLEX(TM) Mixed Integer Optimizer (Version 2.0)*, 1992.

[15] G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, 1963.

[16] R. De Leone and O. L. Mangasarian. Serial and parallel solution of large scale linear programs by augmented Lagrangian successive overrelaxation. In A. Kurzhanski, K. Neumann, and D. Pallaschke, editors, *Optimization, Parallel Processing and Applications*, pages 103–124, Berlin, 1988. Springer–Verlag. Lecture Notes in Economics and Mathematical Systems 304.

[17] R. De Leone and M. A. Tork Roth. Massively parallel solution of quadratic programs via successive overrelaxation. *Concurrency: Practice and Experience*, 5:623–634, 1993.

[18] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39:27–34, 1996.

[19] M. C. Ferris and O. L. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4(4):815–832, 1994.

[20] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[21] Christodoulos A. Floudas. *Nonlinear and Mixed-Integer Optimization : Fundamentals and Applications (Topics in Chemical Engineering.* Oxford Univ Press, New York, 1995.

[22] K. Fukunaga. *Statistical Pattern Recognition.* Academic Press, NY, 1990.

[23] A. A. Gaivoronski. Convergence properties of backpropagation for neural nets via theory of stochastic gradient methods. part i. *Optimization Methods and Software*, 4(2), 1994.

[24] R. S. Garfinkel and G. L. Nemhauser. *Integer Programming.* John Wiley & Sons, New York, 1972.

[25] F. Glover. Improved linear programming models for discriminant analysis. *Decision Sciences*, 21:771–785, 1990.

[26] R. C. Grinold. Mathematical methods for pattern classification. *Management Science*, 19:272–289, 1972.

[27] M. H. Hassoun. *Fundamentals of Artificial Neural Networks.* MIT Press, Cambridge, MA, 1995.

[28] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, California, 1991.

[29] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research.* McGraw-Hill, New York, sixth edition, 1995.

[30] P. J. Huber. Robust estimation of location parameter. *Annals of Mathematical Statistics*, 35:73–101, 1964.

[31] P. J. Huber. *Robust Statistics.* John Wiley, New York, 1981.

[32] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Inc, Englewood Cliffs, NJ, 1988.

[33] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning*, San Mateo, CA, 1994. Morgan Kaufmann.

[34] E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *J. Am. Stat. Assoc.*, 53:457–481, 1958.

[35] Samuel Karlin. *Mathematical Methods and Theory in Games, Programming, and Economics, Volumes 1 and 2.* Dover Publications, Mineola, New York, 1992.

[36] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[37] A. Klarbring. Mathematical programming in contact problems. In M. H. Aliabadi and C. A. Brebbia, editors, *Computational Methods in Contact Mechanics*, pages 233–263. Computational Mechanics Publications, Southampton, England, 1993.

[38] David G. Kleinbaum. *Survival Analysis.* Springer-Verlag, New York, 1996.

[39] Y. le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems II (Denver 1989)*, pages 598–605, San Mateo, California, 1990. Morgan Kaufmann.

[40] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Interior-point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6(1):1–38, 1994.

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

[42] O. L. Mangasarian. *Nonlinear Programming.* McGraw–Hill, New York, 1969. Reprint: SIAM Classic in Applied Mathematics 10, 1994, Philadelphia.

[43] O. L. Mangasarian. Mathematical programming in neural networks. *ORSA Journal on Computing*, 5(4):349–360, 1993.

[44] O. L. Mangasarian. *Nonlinear Programming.* SIAM, Philadelphia, PA, 1994.

[45] O. L. Mangasarian. Machine learning via polyhedral concave minimization. In H. Fischer, B. Riedmueller, and S. Schaeffler, editors, *Applied Mathematics and Parallel Computing - Festschrift for Klaus Ritter*, pages 175–188. Physica-Verlag A Springer-Verlag Company, Heidelberg, 1996. Available by ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-20.ps.Z.

[46] O. L. Mangasarian. Mathematical programming in machine learning. In G. Di Pillo and F. Giannessi, editors, *Nonlinear Optimization and Applications*, pages 283–295, New York, 1996. Plenum Publishing.

[47] O. L. Mangasarian and R. R. Meyer. Nonlinear perturbation of linear programs. *SIAM Journal on Control and Optimization*, 17(6):745–752, November 1979.

[48] O. L. Mangasarian and M. V. Solodov. Serial and parallel backpropagation convergence via nonmonotone perturbed minimization. *Optimization Methods and Software*, 4(2):103–116, 1994.

[49] O. L. Mangasarian, W. Nick Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, July-August 1995.

[50] D. Mladenic. Combinatorial optimization in inductive concept learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 205–211, San MAteo, CA, 1993. Morgan Kaufmann.

[51] P. M. Murphy and D. W. Aha. UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, www.ics.uci.edu/AI/ML/MLDBRepository.html, 1992.

[52] K. G. Murty. *Linear Programming.* John Wiley & Sons, New York, 1983.

[53] K. G. Murty. *Network Programming.* Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[54] K. G. Murty. *Operations Research.* Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[55] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley, New York, 1988.

[56] M. L. Overton. A quadratically convergent method for minimizing a sum of euclidean norms. *Mathematical Programming*, 27:34–63, 1983.

[57] P. D. Panagiotopoulos. *Inequality Problems in Mechanics and Applications.* Birkhäuser, Boston, 1985.

[58] M. R. Rao. Cluster analysis and mathematical programming. *Journal of the American Statistical Association*, 66:622–626, 1971.

[59] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

[60] R. T. Rockafellar. *Network Flows and Monotropic Optimization.* Wiley-Interscience, New York, 1984.

[61] A. Roy, L. S. Kim, and S. Mukhopadhyay. A polynomial time algorithm for the construction and training of a class of multilayer perceptrons. *Neural Networks*, 6:535–545, 1993.

[62] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing.* MIT Press, Cambridge, Massachusetts, 1986.

[63] C. Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10:153–178, 1993.

[64] S. Z. Selim and M. A. Ismail. K-Means-Type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:81–87, 1984.

[65] J. W. Shavlik and T. G. Dietterich (editors). *Readings in Machine Learning.* Morgan Kaufman, San Mateo, California, 1990.

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

[67] W. Nick Street and O. L. Mangasarian. Improved generalization via tolerant training. Technical Report 95-11, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, July 1995. Machine Learning, submitted. Available by ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-11.ps.Z.

[68] Y. Z. Tsypkin. *Foundations of the Theory of Learning Systems.* Academic Press, New York, 1973.

[69] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions.* Kluwer Academic Publishers, Hingham, MA, 1997.

[70] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[71] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, New Jersey, 1944.

[72] W. H. Wolberg, W. N. Street, and O. L. Mangasarian. WPBC: Wisconsin Prognostic Breast Cancer Database. Computer Sciences Department, University of Wisconsin, Madison, ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/WPBC/, 1995.

[73] D. H. Wolpert, editor. *The Mathematics of Generalization*, Reading, MA, 1995. Addison-Wesley.