Mathematical Programming for Data Mining: Formulations and Challenges

P. S. Bradley **Usama M. Fayyad** **O. L. Mangasarian**

Decision Theory & Adaptive Systems
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{bradley,fayyad}@microsoft.com

Computer Sciences Dept.
University of Wisconsin
1210 West Dayton St.
Madison, WI 53706
olvi@cs.wisc.edu

INFORMS *Journal on Computing 11, 1999, 217-238*

January, 1998; Revised July, 1998

Abstract

This paper is intended to serve as an overview of a rapidly emerging research and applications area. In addition to providing a general overview, motivating the importance of data mining problems within the area of knowledge discovery in databases, our aim is to list some of the pressing research challenges, and outline opportunities for contributions by the optimization research communities. Towards these goals, we include formulations of the basic categories of data mining methods as optimization problems. We also provide examples of successful mathematical programming approaches to some data mining problems.

keywords: data analysis, data mining, mathematical programming methods, challenges for massive data sets, classification, clustering, prediction, optimization.

Data Mining and Knowledge Discovery in Databases (KDD) are rapidly evolving areas of research that are at the intersection of several disciplines, including statistics, databases, pattern recognition/AI, optimization, visualization, and high-performance and parallel computing. In this paper, we outline the basic notions in this area, define some of the key ideas and problems, and motivate their importance. One of our goals is to outline areas to which the optimization research community can make significant contributions. Towards this end, we follow our high-level coverage of this area with specific formulations of some of the basic problems in data mining as mathematical programming problems.

To make the exposition concrete, we include case studies where mathematical programming approaches provided effective solutions to data mining problems. However, the bulk of applications in this area have been achieved using statistical pattern recognition, machine learning, and database approaches. We do not cover those application case studies in this paper. Applications are covered in [45], and in a special issue of *Communications of the ACM* [42] (see http://research.microsoft.com/datamine/ACM-contents.htm for contents and abstracts). A new technical journal, *Data Mining and Knowledge Discovery*, dedicated to this area has also been recently launched (see http://research.microsoft.com/datamine for more details).

0.1 From Transactions to Warehouses to KDD

With the widespread use of databases and the explosive growth in their sizes, individuals and organizations are faced with the problem of effectively utilizing this data. Traditionally, "use" of data has been limited to querying a reliable store via some well-circumscribed application or canned report-generating entity. While this mode of interaction is satisfactory for a wide-class of well-defined processes, it was not designed to support data exploration, decision support applications, and ad hoc querying of the data. Now that data capture and storage has become easy and inexpensive, certain questions begin to naturally arise: Will this data help my business gain an advantage? How can we use historical data to build models of underlying processes which generated such data? Can we predict the behavior of such processes? How can we "understand" the data? These questions become particularly important in the presence of massive data sets. A large database represents a large body of information that is presumed to be valuable since it records vital measurements of an entity of interest, be it a business venture, a scientific endeavor, or the operations of a government entity. Yet in a typical setting, this potentially valuable data resource is far from being effectively accessible. The current interfaces between humans and storage systems do not support navigation, exploration, summarization, or modeling of large databases. Providing these types of capabilities and more is the goal of the emerging research area of Data Mining and Knowledge Discovery in Databases.

As transaction processing technologies were developed and became the mainstay of many business processes, great advances in addressing problems of reliable and accurate data capture were achieved. While transactional systems provide a solution to the problem of logging and book-keeping, little emphasis was placed on supporting summarization, aggregation, and ad hoc querying over transactional stores. A recent wave of activity in the database field, called *data warehousing*, has been concerned with turning transactional data into more traditional relational databases that can be queried for summaries and aggregates of transactions. Data warehousing also includes the integration of multiple sources of data along with handling the host of problems associated with such an endeavor. These problems include: dealing with multiple data formats, multiple database management systems (DBMS), integrating distributed databases, data cleaning, and providing a unified logical view of an underlying collection of nonhomogeneous databases.

A data warehouse represents a large collection of data which in principle can provide views of the data that are not practical for individual transactional sources. For example, a supermarket chain may want to compare sales trends across regions at the level of products, broken down by weeks, and by class of store within a region. Such views are often precomputed and stored in special-purpose data stores that provide a multi-dimensional front-end to the underlying relational database and are sometimes called multi-dimensional databases (see [32] for an overview).

Data warehousing is the first step in transforming a database system from a system whose primary purpose is *reliable storage* to one whose primary use is *decision support*. A closely related area is called On-Line Analytical Processing (OLAP), named after principles first advocated by Codd [37]. The current emphasis of OLAP systems is on supporting query-driven exploration of the data warehouse. Part of this entails precomputing aggregates along data "dimensions" in the multi-dimensional data store. Because the number of possible aggregates is exponential in the number of "dimensions", much of the work on OLAP systems is

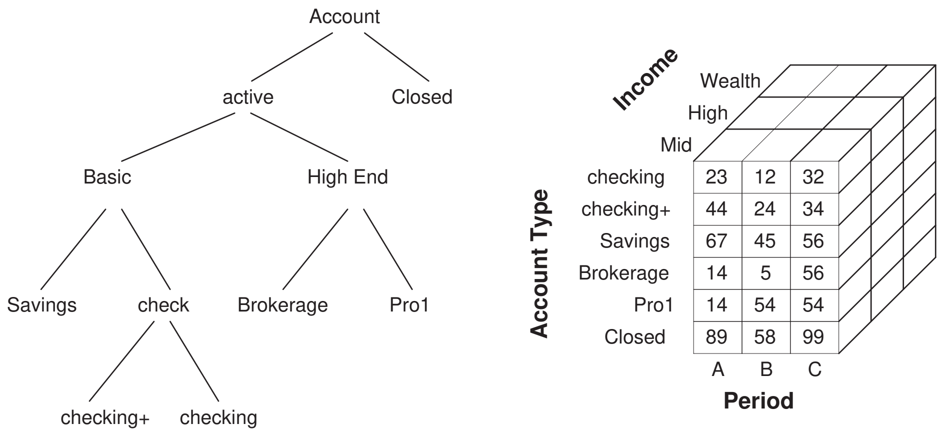


Figure 1: Illustration of the OLAP data cube.

concerned with deciding which aggregates to pre-compute and how to derive other aggregates (or estimate them reliably) from the precomputed projections. Figure 1 illustrates one such example for data representing summaries of financial transactions in branches of a nationwide bank. The attributes (dimensions) have an associated hierarchy which defines how quantities are to be aggregated (rolled-up) as one moves to higher levels in the hierarchy. The entry in each cell in this cube is typically an aggregate of some quantity of interest (e.g. sales) or a count of items (e.g. number of items sold). In the example of Figure 1, our bank recorded 23 transactions on checking accounts during period A made by customers in the "Mid" income range. This cell appears as the top left-hand corner cell of the three dimensional cube illustrated. The hierarchy on the left in this figure allows the user to produce similar summaries at various levels of the hierarchy. The illustration shows the "account type" dimension at the leaves of the hierarchy.

Note that the multidimensional store may not necessarily be materialized as in principle it could be derived dynamically from the underlying relational database. For efficiency purposes, some OLAP systems employ "lazy" strategies in precomputing summaries and incrementally build up a cache of aggregates [32, 61].

0.2 Why Data Mining?

In the OLAP framework, the analysis and exploration is driven entirely by the human analyst. Hence OLAP may be viewed as extending the SQL querying framework to accommodate queries that if executed on a relational DBMS would be computationally prohibitive. Unlike OLAP, data mining techniques allow for the possibility of computer-driven exploration of the data. This opens up the possibility for a new way of interacting with databases: specifying queries at a much more abstract level than SQL permits. It also facilitates data exploration for problems that, due to high-dimensionality, would otherwise be very difficult for humans to solve, regardless of difficulty of use of, or efficiency issues with, SQL.

A problem that has not received much attention in database research is the *query formulation problem*: how can we provide access to data when the user does not know how to describe the goal in terms of a specific query? Examples of this situation are fairly common in decision support situations. For example, in a business setting, say a credit card or telecommunications company would like to query its database of usage data for records representing fraudulent cases. In a data analysis context, a scientist dealing with a large body of data would like to request a catalog of events of interest appearing in the data. Such patterns, while recognizable by human analysts on a case by case basis are typically very difficult to describe in a SQL query. A more natural means of interacting with the database is to state the query by example. In this case, the analyst would label a training set of cases of one class versus another and let the data mining system build a model for distinguishing one class from another. The system can then apply the extracted classifier to search the full database for events of interest. This is typically more feasible because examples are usually easily available, and humans find it natural to interact at the level of cases.

Another major problem which data mining could help alleviate is the fact that humans find it particularly difficult to visualize and understand a large data set. Data can grow along two dimensions: the number of fields (also called dimensions or attributes) and the number of cases. Human analysis and visualization abilities do not scale to high-dimensions and massive volumes of data. A standard approach to dealing with high-dimensional data is to project it down to a very low-dimensional space and attempt to build models in this simplified subspace. As the number of dimensions grow, the number of choice combinations for dimensionality reduction explode. Furthermore, a projection to lower dimensions could easily transform a relatively easy discrimination problem into one that is extremely difficult. In fact, some mining algorithms (e.g. support vector machines [120] discussed later in this paper) employ a reverse techniques where dimensionality is purposefully increased to render the classification problem easy (linear).

However, even if one is to accept that dimensionality reduction is necessary if exploration is to be guided by a human, this still leaves a significant projection selection problem to solve. It is infeasible to explore all of the ways of projecting the dimensions or selecting the right subsamples (reduction along columns versus rows). An effective means to visualize data would be to employ data mining algorithms to perform the appropriate reductions. For example, a clustering algorithm could pick out a distinguished subset of the data embedded in a high-dimensional space and proceed to select a few dimensions to distinguish it from the rest of the data or from other clusters. Hence a much more effective visualization mode could be established: one that may enable an analyst to find patterns or models which may otherwise remain hidden in the high-dimensional space.

Another factor that is turning data mining into a necessity is that the rates of growth of data sets exceed by far any rates with which traditional "manual" analysis techniques could cope. Hence, if one is to utilize the data in a timely manner, it would not be possible to achieve this goal in the traditional data analysis regime. Effectively this means that most of the data would remain unused. Such a scenario is not realistic in any competitive environment where those who better utilize data resources will gain a distinct advantage. This sort of pressure is present in a wide variety of organizations, spanning the spectrum from business, to science, to government. It is leading to serious reconsideration of data collection and analysis strategies that are nowadays causing the accumulation of huge "write-only" data stores. The stores are "write-only" because there are no tools to make access natural, convenient, or easy: the result is no one bothers to read.

1 KDD and Data Mining

The term *data mining* is often used as a synonym for the process of extracting useful information from databases. In this paper, as in [44], we draw a distinction between the latter, which we call KDD, and "data mining". The term *data mining* has been mostly used by statisticians, data analysts, and the database communities. The earliest uses of the term come from statistics and its usage in most settings was negative with connotations of blind exploration of data without *a priori* hypotheses to be verified. However, notable exceptions can be found. For example, as early as 1978 [75], the term is used in a positive sense in a demonstration of how generalized linear regression can be used to solve problems that are very difficult for humans and the traditional statistical techniques of that time to solve. The term KDD was coined at the first KDD workshop in 1989 [101] to emphasize that "knowledge" is the end product of a data-driven process.

In our view KDD refers to the overall *process* of discovering useful knowledge from data while *data mining* refers to a particular *step* in this process. Data mining is the application of specific algorithms for extracting structure from data. The additional steps in the KDD process, such as data preparation, data selection, data cleaning, incorporating appropriate prior knowledge, and proper interpretation of the results of mining, are essential to ensure that useful knowledge is derived from the data. Blind application of data mining methods (rightly criticized as "data dredging" in the statistical literature) can be a dangerous activity easily leading to discovery of meaningless patterns. We give an overview of the KDD process in Figure 2. Note that in the KDD process, one typically iterates many times over previous steps and the process is fairly messy with plenty of experimentation. For example, one may select, sample, clean, and reduce data only to discover after mining that one or several of the previous steps need to be redone. We have omitted arrows illustrating these potential iterations to keep the figure simple.

1.1 Basic Definitions

We adopt the definitions of KDD and data mining provided in [44] as follows:

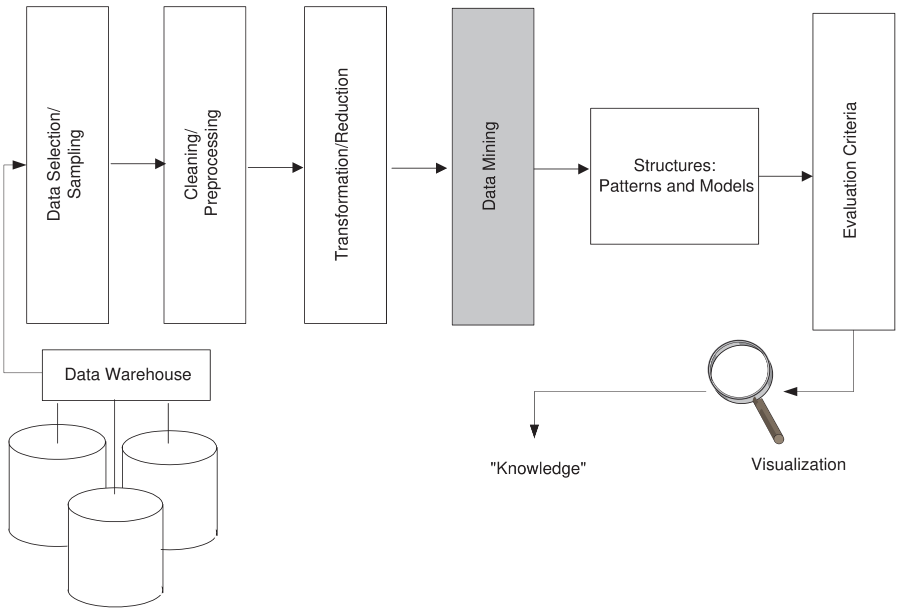


Figure 2: An overview of the steps comprising the KDD process.

Knowledge Discovery in Databases: is *the process of identifying valid, novel, potentially useful, and ultimately understandable structure in data*. This process involves *selecting or sampling data* from a *data warehouse*, *cleaning or preprocessing* it, *transforming or reducing* it (if needed), applying a *data mining component* to produce *structure*, and then *evaluating the derived structure*. See Figure 2.

By *structure* we mean models or patterns. A pattern is classically defined to be a parsimonius description of a subset of data. a model is typically a description of the entire data.

Data Mining: is a *step in the KDD process concerned with the algorithmic means by which patterns or models (structures) are enumerated from the data under acceptable computational efficiency limitations.*

The structure produced by the data mining component must meet certain criteria to be deemed knowledge (the *evaluation criteria* phase of the KDD process (see Figure 2)). Criteria of interest include *validity* (e.g. estimated prediction accuracy on new data) or *utility* (gain, perhaps in dollars saved due to better predictions or speed-up in response time of a system). Other criteria such as *novelty* and *understandability* are much more subjective and difficult to define. In certain contexts understandability can be estimated by simplicity (e.g., the number of bits to describe a pattern). Sometimes the measures are combined under a single *interestingness* measure (e.g., see [111] and references within). Interestingness functions can be explicitly defined or can be manifested implicitly via an ordering placed by the KDD system on the discovered patterns or models. The term knowledge in KDD is user-oriented, domain-specific, and determined by the interestingness measure; it is not a general definition of knowledge and should not be taken as an attempt at a general (e.g. philosophical) definition.

Data mining techniques can be divided into five classes:

1. Predictive Modelling: where the goal is to predict a specific attribute (column or field) based on the other attributes in the data. We discuss this class of techniques in Section 2.

2. Clustering: also called segmentaion, targets grouping the data records into subsets where items in each subset are more "similar" to each other than to items in other subsets. We discuss these thechniques in Section 3).

3. Dependency Modeling: targets modeling the generating joint probability density function of the process (or processes) that could have generated the data. We discuss this class of techniques in Section 4.

4. Data Summarization: targets finding interesting summaries of parts of the data. For example, similarity between a few attributes in a subset of the data.

5. Change and Deviation Detection: accounts for sequence information in data records. Most methods above do not explicitly model the sequence order of items in the data.

The last two items are covered briefly in Section 5. The first three are covered in detail in their respective sections. Many of these techniques have been historically developed to work over memory-resident data, and not much attention has been given to integrating them with database systems. Some of these techniques are beginning to be scaled to operate on large databases. In classification, examples include scalable decision tree algorithms [95, 33] and scalable approaches to computing classification surfaces [21, 102]. In clustering scalable approaches include [17, 125, 60, 2]. In data summarization, examples include [18, 78, 3].

We provide example formulations of some data mining problems as mathematical programs. The formulations are intended as general guidelines and do not necessarily represent best-possible formulations. The goal is to define some of the problems and show how they can be addressed in an optimization framework.

We next describe our notation.

1.2 Notation

- 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 the components $x_j,\ j = 1, \ldots, n$ of x.
- For a vector x in R^n, x_+ denotes the vector in R^n with components $\max\{0, x_i\}$.
- For a vector x in R^n, x_* denotes the vector in R^n with components $(x_*)_i$ equal 1 if $x_i > 0$ and 0 otherwise (i.e. x_* is the result of applying the step function to the components of x).
- The base of the natural logarithm will be denoted by ε, and for a vector $y \in R^m$, ε^{-y} will denote a vector in R^m with components $\varepsilon^{-y_i},\ i = 1, \ldots, m$.
- For $x \in R^n$ and $1 \leq p < \infty$, the norm $\|x\|_p$ will denote the p-norm: $\|x\|_p = \left(\sum_{j=1}^{n} |x_j|^p\right)^{\frac{1}{p}}$ and $\|x\|_\infty = \max_{1 \leq j \leq n} |x_j|$.
- 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 of A and A_i will denote the i-th row of A.
- 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.
- The notation $\arg\min_{x \in S} f(x)$ will denote the set of minimizers of $f(x)$ on the set S.
- 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 $\mathcal{B}$.
- For two vectors $x, y \in R^n$, $x \perp y$ denotes orthogonality, that is the scalar product $x^T y = 0$.

We now discuss the predictive modeling data mining class.

2 Predictive Modeling

The goal of predictive modelling is to estimate a function g which maps points or feature vectors from an input space $\mathcal{X}$ to an output space $\mathcal{Y}$, given only a finite sampling of the mapping, $\{x^i, g(x^i)\}_{i=1}^{M} \subset R^{n+1}$. Hence we are to predict the value of some field ($\mathcal{Y}$) in a database based on the values of other fields ($\mathcal{X}$). We are to accurately construct an estimator $\hat{g}$ of g from this finite sampling or *training set*. The training set may or may not be corrupted by noise.

Problems in predictive modeling can be distinguished by the form of the output space $\mathcal{Y}$. If the predicted quantity is is numeric or continuous (i.e. $\mathcal{Y} = R$, the real line), then the prediction problem is a *regression problem* (e.g. predicting a physical measurement such as height). If the predicted quantity is discrete (i.e. $\mathcal{Y} = \{0, 1, \ldots, K-1\}$) then we have a *classification problem* (e.g. predicting whether a tumor is benign or cancerous).

There are a wide variety of techniques for classification and regression. In general, the problem is cast as determining the most likely value of $\mathcal{Y}$ given the other fields $\mathcal{X}$ over the training data (in which the target variable $\mathcal{Y}$ is given for each observation), and utilizing one's prior knowledge of the problem.

Linear regression combined with non-linear transformation on inputs could be used to solve a wide range of problems. Transformation of the input space is typically the difficult problem requiring knowledge of the problem and quite a bit of "art". In classification problems this type of transformation is often referred to as "feature extraction".

The issue of evaluating the estimate $\hat{g}$ in terms of how well it performs on data not included in the training set, or how well $\hat{g}$ generalizes, is paramount. Often it is possible to allow an algorithm to construct $\hat{g}$ from a sufficiently complex function class so that $\hat{g}$ approximates g arbitrarily well on the training set. But this complex $\hat{g}$ usually approximates g poorly on points not in the training set [110]. This is the case of *overfitting* the training data. While biasing a classification algorithm to construct $\hat{g}$ from a less complex function class often improves generalization ability, it may not be desirable in all problem domains [106]. *Overtraining* can also lead to poor generalization even when the complexity of the function class from which $\hat{g}$ is constructed is optimal [14]. The key to good generalization is correctly estimating the complexity of the true mapping g while avoiding overtraining. This problem is compounded by the fact that we have only a finite sampling of g, which, in addition, may be corrupted by noise.

In general, it should be noted that the problem of trading off the simplicity of a model with how well it fits the training data is a well-studied problem. In statistics this is known as the *bias-variance tradeoff* [54], in Bayesian inference it is known as *penalized likelihood* [13, 63], and in pattern recognition/machine learning it manifests itself as the *minimum message length* (MML) [123] problem. The MML framework, also called minimum description length (MDL) [103] dictates that the best model for a given data set is one that minimizes the coding length of the data and the model combined. If a model fits the data exactly, then the data need not be encoded and the cost is that of coding the model. Similarly, if the data is not represented by a model, then the cost of this scheme is the cost of encoding the data. One can show that minimizing the MDL is equivalent to selecting the model that minimizes the Bayes risk assuming cost of errors is uniform, specifically, for a data set D, the MDL prefers the model M for which $\text{Prob}(M|D)$ is maximized. This can be shown by applying Bayes rule:

$$\text{Prob}(M|D) = \text{Prob}(D|M)\frac{\text{Prob}(M)}{\text{Prob}(D)}$$

and then taking the logarithm of each side. This reduces to

$$-\log(\text{Prob}(M|D)) = -\log(\text{Prob}(D|M)) - \log(\text{Prob}(M)) + log(\text{Prob}(D))$$

Noting that $\text{Prob}(D)$ is a constant for all models being compared, and that the minimal cost of encoding an object requires at least logarithm of its probability in bits, we see that MDL calls for choosing the model with the maximum likelihood given the data.

Given this brief introduction to predictive modelling we focus our attention on the classification problem.

2.1 Classification

2.1.1 Overview

In classification the basic goal is to predict the most likely state of a categorical variable (the class) given the values of other variables. This is fundamentally a density estimation problem. If one could estimate the probability that the class (value of $\mathcal{Y}$), given the value of $x \in \mathcal{X}$, then one could derive this probability from the joint density on $\mathcal{Y}$ and $\mathcal{X}$. However, this joint density is rarely known and difficult to estimate. Hence one has to resort to various techniques for estimating this density, including:

1. Density estimation, e.g. kernel density estimators [41] or graphical representations of the joint density [63].
2. Metric-space based methods: define a distance measure on data points and guess the class value based on proximity to data points in the training set. For example, the K-nearest-neighbor method [41].
3. Projection into decision regions: divide the attribute space into decision regions and associate a prediction with each. For example linear discriminant analysis determines linear separators and neural networks compute non-linear decision surfaces [64]. Decision tree or rule-based classifiers make a piecewise constant approximation of the decision surface [26, 81, 6].

The third class of methods is by far the most commonly used and studied. It is usually more practical because it sidesteps the harder problem of determining the density and just concentrates on separating various regions of the space.

2.1.2 Mathematical Programming Formulations

We address the task of estimating a classification function which assigns a given vector $x \in R^n$ into one of two disjoint point sets $\mathcal{A}$ or $\mathcal{B}$ in n-dimensional feature space. We have $\mathcal{X} = R^n$, $\mathcal{Y} = \{0, 1\}$ and the classification function has the following form.

$$g(x) = \begin{cases} 1 & \text{if } x \in \mathcal{A} \\ 0 & \text{if } x \in \mathcal{B}. \end{cases} \tag{1}$$

We represent the m elements of the finite point set $\mathcal{A} \subset R^n$ as the matrix $A \in R^{m \times n}$ where each element of $\mathcal{A}$ is represented by a row in A. Similarly, we represent the k elements of the finite point set $\mathcal{B}$ as $B \in R^{k \times n}$.

We attempt to discriminate between the points of $\mathcal{A}$ and $\mathcal{B}$ by constructing a separating plane:

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

with normal $w \in R^n$ and distance $\dfrac{|\gamma|}{\|w\|_2}$ to the origin. We wish to determine w and γ so that the separating plane P defines two open halfspaces $\{x \mid x \in R^n, x^T w > \gamma\}$ containing mostly points of $\mathcal{A}$, and $\{x \mid x \in R^n, x^T w < \gamma\}$ containing mostly points of $\mathcal{B}$. Hence we wish to satisfy

$$Aw > e\gamma, \ Bw < e\gamma \tag{3}$$

to the extent possible. Upon normalization, these inequalities can be equivalently written as follows:

$$Aw \geq e\gamma + e, \ Bw \leq e\gamma - e. \tag{4}$$

Conditions, (3) or equivalently (4), can be satisfied if and only if, the convex hulls of $\mathcal{A}$ and $\mathcal{B}$ are disjoint. This is not the case in many real-world applications. Hence, we attempt to satisfy (4) in some "best" sense, for example, by minimizing some norm of the average violations of (4) such as

$$\min_{w,\gamma} f(w, \gamma) = \min_{w,\gamma} \frac{1}{m} \|(-Aw + e\gamma + e)_+\|_1 + \frac{1}{k} \|(Bw - e\gamma + e)_+\|_1. \tag{5}$$

Recall that for a vector x, x_+ denotes the vector with components $\max\{0, x_i\}$.

Two principal reasons for choosing the 1-norm in (5) are:

(i) Problem (5) is then reducible to a linear program (6) with many important theoretical properties making it an effective computational tool [9].

(ii) The 1-norm is less sensitive to outliers such as those occurring when the underlying data distributions have pronounced tails, hence (5) has a similar effect to that of robust regression [65],[62, pp 82-87].

The formulation (5) is equivalent to the following robust linear programming formulation (RLP) proposed in [8] and effectively used to solve problems from real-world domains [91]:

$$\min_{w,\gamma,y,z} \left\{ \frac{e^T y}{m} + \frac{e^T z}{k} \,|\, -Aw + e\gamma + e \leq y,\ Bw - e\gamma + e \leq z,\ y \geq 0,\ z \geq 0 \right\}. \tag{6}$$

The linear program (6) or, equivalently, the formulation (5) define a separating plane P that approximately satisfies the conditions (4). We note that this LP is feasible ($w = 0$, $\gamma = 0$, $y = e$, $z = e$) and the objective is bounded below by zero, hence a solution to (6) always exists. We further note that the solution $w = 0$ occurs if and only if

$$\frac{e^T A}{m} = \frac{e^T B}{k}, \tag{7}$$

in which case the solution $w = 0$ is not unique [9, Theorems 2.5 and 2.6]. Hence a useful plane P (2) is always generated by the robust linear program (6).

The linear programming formulation (6) obtains an approximate separating plane that minimizes a weighted sum of the distances of misclassified points to the approximate separating plane. Minimization of such a weighted sum of the distances of misclassified points by a linear program is merely a surrogate for minimizing the number of misclassified points by the separating plane. Next, we propose a precise mathematical programming formulation of the nonconvex problem of minimizing the number of such misclassified points [83]. This is done by first proposing a simple linear complementarity formulation of the step function (Lemma 2.1) and then using this result to formulate the misclassification minimization problem as a linear program with equilibrium (linear complementarity) constraints (LPEC).

A linear program with equilibrium constraints (LPEC) is a linear program with a single complemetarity constraint (an orthogonality condition between two linear functions). LPECs arise when the constraints involve another linear programming problem. LPECs can model machine learning problems [83, 86], while more general mathematical programs with equilibrium constraints (MPECs) [79] can model more general problems such as economic and traffic equilibrium problems.

Note first that the system (4) is equivalent to

$$e^T(-Aw + e\gamma + e)_* + e^T(Bw - e\gamma + e)_* = 0, \tag{8}$$

where as stated earlier x_* denotes the vector with components $(x_*)_i$ equal to 1 if $x_i > 0$ and 0 otherwise. The left hand side of (8) counts the number of points misclassified by the plane P (2). For the linearly separable case, no points are misclassified and equality in (8) holds. In the more general case where the sets $\mathcal{A}$ and $\mathcal{B}$ have intersecting convex hulls and hence are not linearly separable, equality in (8) does not hold but it is used as a target for the minimization problem

$$\min_{w,\gamma} e^T(-Aw + e\gamma + e)_* + e^T(Bw - e\gamma + e)_*. \tag{9}$$

This problem has a zero minimum if and only if the plane P (2) strictly separates $\mathcal{A}$ from $\mathcal{B}$. Otherwise P minimizes the number of misclassified points. Specifically, it minimizes

$$c(w,\gamma) = \text{cardinality} \left\{ (i,j) \;\middle|\; \begin{array}{c} A_i w - \gamma - 1 < 0, \\ -B_j w + \gamma - 1 < 0, \\ 1 \leq i \leq m, \\ 1 \leq j \leq k \end{array} \right\}. \tag{10}$$

We note that (9) is always solvable since there exists only a finite number of twofold partitions of $\mathcal{A} \cup \mathcal{B}$ that are linearly separable [83]. Any such partition that minimizes the number of points misclassified also minimizes (9). We reduce (9) to an LPEC by first representing the step function $(\cdot)_*$ as a complementarity condition via the plus function $(\cdot)_+$, which is a re-statement of [86, Equation 2.11].

Lemma 2.1 Characterization of the step function $(\cdot)_*$. *For $r \in R^m, u \in R^m, a \in R^m$ and e, a vector of ones in R^m:*

$$r = (a)_*, u = (a)_+ \;\Leftrightarrow\; (r,u) \in \arg\min_{r,u} \left\{ e^T r \;\middle|\; \begin{array}{l} 0 \leq r \perp u - a \geq 0, \\ 0 \leq u \perp -r + e \geq 0 \end{array} \right\}. \tag{11}$$

In the above problem, the complementarity terminology specifically applies to the first set of constraints $0 \leq r \perp\geq 0$ in the following way. If a component of r, say r_i, is strictly positive, then the fact that $u - a \geq 0$ and $r \perp u - a$ implies that $u - a = 0$. In essence, if a component of a vector on one side of the $\perp$ is strictly positive, the corresponding component of the vector on the other side of the $\perp$ must be zero.

We now combine Lemma 2.1 and the minimization problem (9) to obtain the following misclassification minimization characterization which is a re-statement of [83, Proposition 2.2].

Proposition 2.2 Misclassification Minimization as a Linear Program with Equilibrium Constraints (LPEC). *A plane $w^T x = \gamma$ minimizes the number of misclassifications $c(w,\gamma)$ as defined by (10) if and only if (w,γ,r,u,s,v) solve the following linear program with equilibrium constraints:*

$$\begin{array}{lll} \underset{w,\gamma,r,u,s,v}{\text{minimize}} & \multicolumn{2}{c}{e^T r + e^T s} \\ & u + Aw - e\gamma - e \geq 0 & v - Bw + e\gamma - e \geq 0 \\ & r \geq 0 & s \geq 0 \\ \text{subject to} & r^T(u + Aw - e\gamma - e) = 0 & s^T(v - Bw + e\gamma - e) = 0 \\ & -r + e \geq 0 & -s + e \geq 0 \\ & u \geq 0 & v \geq 0 \\ & u^T(-r + e) = 0 & v^T(-s + e) = 0 \end{array} \tag{12}$$

Since problem (12) is a linear program with equilibrium constraints (LPEC), it is a special case of the more general mathematical program with equilibrium constraints (MPEC) studied in detail in [79]. Being linear, (12) is endowed with some features not possessed by the more general MPECs, principally exactness of a penalty formulation without boundedness of the feasible region and without assuming nondegeneracy. These properties and an algorithm for solving (12) are given in [83].

We note that a hybrid algorithm is proposed in [35] addressing the misclassification minimization problem (9) which performs the following two steps at each iteration. The first step consists of determining $w \in R^n$ by solving the linear program (6) for a fixed value of $\gamma \in R^n$. The new γ is then determined as the one that minimizes the number of points misclassified by the separating plane P, with w determined in the previous step. The Hybrid Misclassification Minimization Algorithm [35, Algorithm 5] terminates when the number of misclassifications is not decreased.

Up to this point we have proposed mathematical programming formulations for computing a separating plane (2) to distinguish between the members of $\mathcal{A}$ and $\mathcal{B}$ in the the training set. We have not yet addressed the issue of *generalization* or how well the estimated classification function $\hat{g}$ (1) performs on new data not included in the training set. We investigate two approaches with the goal of improving the generalization ability of the resulting classifier. The first is *feature selection* or computing a separating surface utilizing a minimum number of problem features. The second is the *support vector machine* which attempts to compute a separating with a maximum margin of separation between the two sets $\mathcal{A}$ and $\mathcal{B}$.

2.1.3 Feature Selection

We propose improving the generalization ability of $\hat{g}$ (1) by computing a separating plane P which utilizes a minimum number of problem features. Having a minimal number of features often leads to better generalization and simpler models that can be more easily interpreted. This problem is also addressed by statistics [55, 73], machine learning [69, 74] as well as by other mathematical programming formulations [25, 85].

We propose a mathematical programming feature selection approach by introducing an extra term which penalizes the nonzero elements of the weight vector w, with parameter $\lambda \in [0, 1)$ into the objective of (6) while weighting the original objective by $(1 - \lambda)$ as follows:

$$\min_{w,\gamma,y,z} \left\{ (1-\lambda)\left(\frac{e^T y}{m} + \frac{e^T z}{k}\right) + \lambda e^T |w|_* \;\middle|\; \begin{array}{c} -Aw + e\gamma + e \leq y, \\ Bw - e\gamma + e \leq z, \\ y \geq 0, z \geq 0 \end{array} \right\}, \lambda \in [0, 1). \tag{13}$$

Notice that the vector $|w|_* \in R^n$ has components which are equal to 1 if the corresponding components of w are nonzero and components equal to zero if the corresponding components of w are zero. Hence $e^T|w|_*$ counts the number of nonzero components of w. Problem (13) balances the error in separating the sets $\mathcal{A}$ and $\mathcal{B}$, $\left(\frac{e^T y}{m} + \frac{e^T z}{k}\right)$ and the number of nonzero elements of w, $(e^T|w|_*)$. Further, note that if an element of w is zero, the corresponding feature is removed from the problem.

Problem (13) is equivalent to the following parametric program:

$$\min_{w,\gamma,y,z,v} \left\{ (1-\lambda)\left(\frac{e^T y}{m} + \frac{e^T z}{k}\right) + \lambda e^T v_* \;\middle|\; \begin{array}{c} -Aw + e\gamma + e \leq y, \\ Bw - e\gamma + e \leq z, \\ y \geq 0, z \geq 0, \\ -v \leq w \leq v \end{array} \right\}, \lambda \in [0, 1). \tag{14}$$

This feature selection problem will be solved for a value of $\lambda \in [0, 1)$ for which the resulting classification function estimate $\hat{g}$ (1) generalizes best, estimated by cross-validation [114]. Typically this will be achieved in a feature space of reduced dimensionality.

The discontinuous term $e^T v_*$ in (14) can be modelled using Lemma 2.1 resulting in an LPEC formulation [24]. Another approach is to approximate the discontinuous step function with the continous sigmoid function [24]. A further continous approximation utilizes the negative exponential:

$$v_* \approx t(v, \alpha) = e - \varepsilon^{-\alpha v}, \alpha > 0, \tag{15}$$

and leads to the smooth problem:

$$\min_{w,\gamma,y,z,v} \left\{ (1-\lambda)\left(\frac{e^T y}{m} + \frac{e^T z}{k}\right) + \lambda e^T (e - \varepsilon^{-\alpha v}) \;\middle|\; \begin{array}{c} -Aw + e\gamma + e \leq y, \\ Bw - e\gamma + e \leq z, \\ y \geq 0, z \geq 0, \\ -v \leq w \leq v \end{array} \right\}, \lambda \in [0, 1). \tag{16}$$

We note that this problem is the minimization of a concave objective function over a polyhedral set. Even though it is difficult to find a global solution to this problem, the Successive Linear Approximation (SLA) algorithm [24, Algorithm 2.1] terminates at a stationary point which satisfies the minimum principle necessary optimality condition for problem (16) [24, Theorem 2.2]. This algorithm computes solutions which are locally optimal. Even though global optimality cannot be guaranteed, this fast finite procedure produces a sparse weight vector w with empirically observed good generalization properties [24].

2.1.4 Support Vector Machines

The previous discussion of feature selection naturally leads to a strongly related framework: the support vector machine (SVM) [120, 107, 99]. While most approaches are based on the idea of minimizing an error in separating the given data (i.e. minimizing training set error), SVMs incorporate *structured risk minimization* [120, 28] which minimizes an upper bound on the generalization error. For a more detailed discussion of SVMs, see [120, 29, 122].

Consider the simple case when the sets $\mathcal{A}$ and $\mathcal{B}$ are linearly separable. The idea is to determine, among the infinite number of planes correctly separating $\mathcal{A}$ from $\mathcal{B}$, the one which will have smallest generalization error. SVMs choose the plane which maximizes the margin separating the two classes. Margin is defined as the distance between the separating hyperplane to the nearest point in $\mathcal{A}$ plus the distance from the hyperplane to the nearest point in $\mathcal{B}$. Recall that in the linearly separable case, the inequalities (4) are satisfied. The nearest point in $\mathcal{A}$ to the separating plane, denoted as A_i, satisfies $A_iw = \gamma + 1$. Similarly, the nearest point in $\mathcal{B}$ to the separating plane satifies $B_iw = \gamma - 1$. The margin then is $\frac{2}{\|w\|_2}$. If $\mathcal{A}$ and $\mathcal{B}$ are linearly inseparable, SVMs determine a separating plane P that maximizes the margin and minimizes a quantity measuring misclassification errors. In keeping with the notation already introduced, the margin term is weighted by $\lambda \in [0, 1)$ and a measure of misclassification error is weighted by $(1 - \lambda)$:

$$\begin{array}{ll} \underset{w,\gamma,y,z}{\text{minimize}} & (1-\lambda)(e^Ty + e^Tz) + \frac{\lambda}{2}\|w\|_2^2 \\ & -Aw + e\gamma + e \leq y, \\ \text{subject to} & Bw - e\gamma + e \leq z, \\ & y \geq 0, z \geq 0. \end{array} \tag{17}$$

The Wolfe dual [84] of this quadratic programming problem is usually solved. Points $A_i \in \mathcal{A}$ and $B_i \in \mathcal{B}$ corresponding to inequality constraints of (17) with positive dual variables constitute the *support vectors* of the problem. These points are the *only* data points that are relevant for determining the optimal separating plane. Their number is usually small and it is proportional to the generalization error of the classifier [99].

If the margin is measured by some arbitrary norm $\|\cdot\|$, then the term appearing in the objective of the SVM problem penalizes the weight vector with the *dual norm* $\|\cdot\|'$ [87]. For a general norm $\|\cdot\|$ on R^n, the *dual norm* $\|\cdot\|'$ on R^n is defined as

$$\|x\|' = \max_{\|y\|=1} x'y.$$

A p-norm and q-norm are dual to each other for $1 \leq p,\ q \leq \infty$ and $\frac{1}{p} + \frac{1}{q} = 1$. Note that the 2-norm is dual to itself, hence it's appearance in the objective of (17). The 1-norm and ∞-norm are dual to each other. Hence, if the margin is measured in the ∞-norm, the penalty on w is $\|w\|_1$ giving rise to the following linear programming formulation of the SVM problem [20].

$$\begin{array}{ll} \underset{w,\gamma,y,z}{\text{minimize}} & (1-\lambda)(\frac{e^Ty}{k} + \frac{e^Tz}{m}) + \lambda\|w\|_1 \\ & -Aw + e\gamma + e \leq y, \\ \text{subject to} & Bw - e\gamma + e \leq z, \\ & y \geq 0, z \geq 0. \end{array} \tag{18}$$

The linear programming formulations (6), (18) and the linear programming subproblems of the Successive Linearization Approximation algorithm for problem (16) [24] are scaled to massive datasets via the Linear Program Chunking (LPC) algorithm [21]. The quadratic programming formulation of the support vector machine problem is efficiently scaled via the Sequential Minimal Optimization (SMO) algorithm [102] and by "chunking" methods [98].

These mathematical programming formulations have been extended to constructively training neural networks [82, 8, 11, 19], decision tree construction [6, 11, 82, 7] and calculating nonlinear discriminants by nonlinearly transforming the given data [120, 29, 99].

We now present a few case studies.

2.1.5 Case Study: Breast Cancer Diagnosis

The formulation (6) has been successfully used to determine a classification function that objectively performs breast cancer diagnosis [91]. Specifically, cellular features of a breast mass are used to classify the mass as either benign or malignant. This classification function is a central component of the Xcyt image analysis program. This case study is detailed in [91] and is briefly summarized here.

First, a fluid sample from the breast mass is obtained by an outpatient procedure involving a small gauge needle, known as a fine needle aspirate (FNA). The fluid is placed on a slide and stained to highlight the cellular nuclei of the constituent cells. A digitized image is then obtained from the slide.

Xcyt then uses a curve-fitting program to determine exact boundaries of the nuclei, initiated by an operator using a mouse pointer. Ten features are then computed for each nucleus. The mean value, extreme value and standard error are then computed over the nuclei in a given image, for each of the ten features. Hence, this procedure maps each breast mass to a 30-dimensional real-valued vector.

The Wisconsin Diagnostic Breast Cancer Database contains 569 30-dimensional vectors computed by the Xcyt system (available from UCI Repository of Machine Learning Databases, University of California, Irvine, http://www.ics.uci.edu/ ˜mlearn/ MLRepository.html). Actual diagnostic outcome for these 569 patients is known. Malignant cases were verified by biopsy and benign cases confirmed by biopsy or subsequent examinations. The diagnostic tags determine the sets $\mathcal{A}$ and $\mathcal{B}$ for the classification problem.

Best results, determined by 10-fold cross-validation [114], were obtained with a separating plane calculated by (6) using 3 of the 30 dimensions: extreme area, extreme smoothness and mean texture. Predicted tenfold cross-validation accuracy was 97.5%. This level of accuracy is as good as the best results achieved at specialized cancer institutions [91].

2.1.6 Case Study: Face Detection

We present a "Support Vector Machine approach for detecting vertically oriented and unoccluded frontal views of human faces in grey level images" [99]. This case study is detailed in [99] and is briefly summarized here.

In the face detection problem, input is an arbitrary image. The task is then to determine whether or not there are any human faces in the image. If so, return their location. The system discussed here works by scanning an image for candidate face-like patterns at many different scales. A SVM is then used to classify these patterns as face/non-face. The SVM problem that is solved is quadratic programming problem (17).

An SVM is trained from a database of face/non-face $19 \times 19 = 361$ pixel patterns. These 361 dimensions are augmented with quadratic features, hence the computed separating surface is quadratic in the original space of 361 features. The value of λ in (17) was $\frac{200}{201} \approx 0.99502$. In order to compensate for sources in image variation, the following pre-processing steps were performed: masking, illumination gradient correction and histogram equalization. After a decision surface has been obtained through solving problem (17), the run-time system is used over images that do not contain faces to produce negative training examples to use in the future. After re-solving (17), the classifier is incorporated into a run-time system which performs the following operations: 1) re-scale the input image several times, 2) cut 19×19 window patterns, 3) pre-process the window pattern, 4) classify the image, and 5) if the pattern is a face, draw a rectangle around it.

Experimental results presented in [99] used two sets of images. Set A contained 313 high-quality images with one face per image. Set B contained 23 images of mixed quality with a total of 155 faces. "... it is important to state that set A involved 4,669,960 pattern windows, while set B 5,383,682." [99]. The system is compared with [116] in Table 1.

	Test Set A		Test Set B	
	Detect Rate	False Alarms	Detect Rate	False Alarms
SVM	97.1%	4	74.2%	20
Sung *et al.*	94.6%	2	74.2%	11

Table 1: Face detection experimental results.

We now concentrate on the regression task.

2.2 Regression

We now discuss the regression problem which differs from the classification problem in that the function g which we are trying to estimate has a continuous output in contrast to a discrete output for the classification problem. Other methods addressing the regression problem, which are also optimization problems, include neural networks [64] and smoothing splines [121].

2.2.1 Mathematical Programming Formulations

We present mathematical programming formulations that attempt to estimate the true, unknown regression function g by a linear combination of functions from some pre-defined function set. We relax the notion that elements of the function set be linearly independent. Hence the solution to the problem is not unique and from all possible solutions, we wish to find one which performs best, or generalizes, to new data. For instance, the set of functions discussed in [36] consist of an "overcomplete waveform dictionary – stationary wavelets, wavelet packets, cosine packets, chirplets, and warplets, ...". Notice that by choice of the function set, linear regression and polynomial regression are easily cast in this framework.

We are provided with a finite number of samples of the true regression function $\{x^i, g(x^i)\}_{i=1}^M \in R^{n+1}$. We allow for the possibility that g cannot be sampled precisely in which case our given data looks like $\{x^i, g(x^i) + \epsilon^i\}_{i=1}^M$ where ϵ^i is the error in measurement of $g(x^i)$. The set $\mathcal{F} = \{f_1, f_2, \ldots, f_N\}$ denotes the predefined function set with which to form our estimate:

$$\hat{g}(x) = \sum_{j=1}^{N} w_j f_j(x). \tag{19}$$

By sampling the elements of $\mathcal{F}$ at the data points $x^1, x^2, \ldots, x^M$, the problem of estimating the coefficients w_j in (19) reduces to solving the linear system:

$$Aw = b. \tag{20}$$

Here $(A)_{ij} = f_j(x^i)$ and $b_i = g(x^i)$ (or $g(x^i) + \epsilon^i$ in the case involving noise).

Consider the simplest case where there is no noise in the RHS vector b (i.e. g can be sampled precisely), and the set of function in $\mathcal{F}$ gives rise to a matrix A for which exact solutions to $Aw = b$ exist. The method of frames [39] finds a solution with minimal 2-norm:

$$\begin{array}{ll} \underset{w}{\text{minimize}} & \|w\|_2 \\ \text{subject to} & Aw = b. \end{array} \tag{21}$$

For the same, noise-free situation, the Basis Pursuit method [36] finds a solution with minimal 1-norm:

$$\begin{array}{ll} \underset{w}{\text{minimize}} & \|w\|_1 \\ \text{subject to} & Aw = b. \end{array} \tag{22}$$

Consider a noisy RHS vector b in which case an exact solution to the system $Aw = b$ may not exist. Then the least squares solution to (20) is obtained by solving the following problem [66]:

$$\min_{w \in R^N} \|Aw - b\|_2. \tag{23}$$

If the elements of b consist of noisy measurements of g and the noise is uncorrelated over samples with zero mean and equal variance, then the estimate computed by solving (23) has the smallest variance in the class of estimation methods satisfying the following two conditions: 1) the estimate is unbiased and, 2) the estimate is a linear function of b [66].

We consider the least one norm formulation which may provide an estimate $\hat{g}$ (19) which is more robust to outliers in the case of noise in b. This formulation is the following linear program

$$\begin{array}{ll}\underset{w,y}{\text{minimize}} & e^T y \\ \text{subject to} & -y \le Aw - b \le y.\end{array} \tag{24}$$

We now explicitly consider determining an estimate $\hat{g}$ from the over-determined function set $\mathcal{F}$ with good generalization properties. We first consider the following Parsimonious Least Norm Approximation (PLNA) which "chooses" the fewest elements of $\mathcal{F}$ such that $\hat{g}$ approximates our sampling of the true regression function [22]:

$$\min_{w}(1-\lambda)\|Aw-b\|_1 + \lambda(e^T|w|_*),\ \lambda \in [0,1). \tag{25}$$

The value of λ is chosen to maximize a measure of generalization, estimated by cross-validation [114], for instance.

Problem (25) can be re-formulated as the following nondifferentiable constrained optimization problem for $\lambda \in [0,1)$, (in all further problem formulations, λ will be assumed to be in this interval):

$$\begin{array}{ll}\underset{w,y,v}{\text{minimize}} & (1-\lambda)(e^T y) + \lambda(e^T v_*) \\ \text{subject to} & \begin{array}{c}-y \le Aw - b \le y, \\ -v \le w \le v\end{array}\end{array} \tag{26}$$

The step vector v_* can again be approximated by the concave exponential (15). With this approximation (25) becomes the following concave minimization over a polyhedral set:

$$\begin{array}{ll}\underset{w,y,v}{\text{minimize}} & (1-\lambda)(e^T y) + \lambda(e^T(e - \varepsilon^{-\alpha v})) \\ \text{subject to} & \begin{array}{c}-y \le Aw - b \le y, \\ -v \le w \le v\end{array}\end{array} \tag{27}$$

This problem is effectively solved by the successive linearization algorithm [22, Algorithm 3.1] which terminates finitely at a point satisfying the minimum principle necessary optimality condition for (27). For a sufficiently large, but finite value of the parameter α in the negative exponential, a solution of (27) is in fact a solution of (26) [22, Theorem 2.1].

The Least Least Norm Approximation (LLNA) method minimizes the 1-norm residual plus a 1-norm penalty on the size of the coefficient vector w [22] whose solution involves solving a single linear program:

$$\min_{w}(1-\lambda)\|Aw-b\|_1 + \lambda\|w\|_1. \tag{28}$$

The Method-of-Frames De-Noising [36] refers to the minimization of the least square fit error plus a 2-norm penalization term on the coefficient vector:

$$\min_{w} \frac{(1-\lambda)}{2}\|Aw-b\|_2^2 + \frac{\lambda}{2}\|w\|_2^2. \tag{29}$$

Similarly, the Basis Pursuit De-Noising [36] refers to a solution of:

$$\min_{w} \frac{(1-\lambda)}{2}\|Aw-b\|_2^2 + \lambda\|w\|_1. \tag{30}$$

The Support Vector Method has also been extended to the regression problem [119]. In our setting, this problem can then be formulated as:

$$\min_{w} \frac{(1-\lambda)}{N}(e^T(|Aw-b|_\epsilon) + \lambda\|w\|_2^2. \tag{31}$$

Here $|\cdot|_\epsilon$ is the following loss functional [119]:

$$|\zeta|_\epsilon = \begin{cases} 0 \text{ if } |\zeta| < \epsilon, \\ |\zeta| - \epsilon \text{ otherwise.} \end{cases} \tag{32}$$

Problem (31) can then be formulated as the following constrained quadratic program:

$$\begin{array}{ll} \underset{w,y}{\text{minimize}} & \frac{(1-\lambda)}{N}(e^T y) + \lambda(w^T w) \\ \text{subject to} & -y - \epsilon e \le Aw - b \le y + \epsilon e, \\ & y \ge 0. \end{array} \tag{33}$$

In [115] a similar Tolerant Training Method is proposed in which the following quadratic program is solved for some nonnegative tolerance τ (which is analogous to ϵ above) and some small value of $\frac{\lambda}{1-\lambda}$:

$$\begin{array}{ll} \underset{w,y,z}{\text{minimize}} & \frac{(1-\lambda)}{2}\left[\|y\|_2^2 + \|z\|_2^2\right] + \frac{\lambda}{2}\|w\|_2^2 \\ \text{subject to} & -z - \tau e \le Aw - b \le y + \tau e, \\ & y, z \ge 0. \end{array} \tag{34}$$

Again, the linear programming formulation (28) and the linear programming subproblems of the Successive Linearization Algorithm for (27) [22] are effectively scaled to massive datasets via the Linear Program Chunking algorithm [21].

2.2.2 Regression Case Study

We now present a case study in attempting to recover a true continuous signal $g(t)$, given only a finite, noisy sampling. This case study appears in [22] and is summarized here.

The linear systems used are based upon ideas related to signal processing [57, 117] and more specifically to an example in [1, Equation (8)].

We consider the following *true* signal $g(t) : [0,1] \longrightarrow R$:

$$g(t) = \sum_{j=1}^{3} x_j \varepsilon^{-a_j t},\ t \in [0,1],\ a = [0\ \ 4\ \ 7]',\ x = [0.5\ \ 2.0\ \ -1.5]'. \tag{35}$$

We assume that the true signal $g(t)$ cannot be sampled precisely, but that the following *observed* signal can be sampled:

$$\tilde{g}(t) = (g(t) + \text{error}), \quad \text{sampled at times}:\ t_i = i \bigtriangleup t,\ \bigtriangleup t = 0.04,\ i = 0, 1, \ldots, 25. \tag{36}$$

We further assume that we do not know the true signal $g(t)$ (35), and we attempt to model it as:

$$\hat{g}(t) = \sum_{j=1}^{10} x_j \varepsilon^{-a_j t},\ t \in [0,1],\ a = [0\ \ 4\ \ 7\ \ 0.1\ \ 2\ \ 3\ \ 3.9\ \ 4.1\ \ 6.9\ \ 7.1]'. \tag{37}$$

The problem now is to compute the coefficients $x_j,\ j = 1, \ldots, 10$, of $\hat{g}(t)$ (37) so that we can adequately recover $g(t)$, given only the noisy data $\tilde{g}(t_i)$ (36). The coefficient vector on the "basis functions" will be denoted by x instead of w, as it is denoted in the problem formulations in Section 2.2. Notice that by substituting the following coefficient vector x^* into (37), $\hat{g}(t) = g(t)$:

$$x^* = [0.5 \;\; 2.0 \;\; -1.5 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]'. \tag{38}$$

Thus the *true* linear system (testing set) $Ax = b$ is then given by:

$$A_{ij} = \varepsilon^{-a_j t_i}, \; b_i = g(t_i), \; i = 0, \ldots, 25, \; j = 1, \ldots 10, \tag{39}$$

and is solved exactly by x^* of (38).

The *observed* linear system (training set) $Ax = b + p$ is then given by:

$$\left\langle \begin{array}{l} A_{ij} = \varepsilon^{-a_j t_i}, \;\; b_i = g(t_i), \\ p_i = \text{random number with mean } = 0 \; \& \text{ standard deviation } = 1, \\ i = 0, \ldots, 25, \; j = 1, \ldots, 10. \end{array} \right\rangle \tag{40}$$

We will refer to a solution of the Parsimonious Least Norm Approximation (25), with b of (25) replaced by $b + p$, as a PLNA solution. Similarly, we shall refer to a solution of problem (28), with b replaced by $b + p$ as an LLNA (Least Least Norm Approximation) solution.

We compute solutions of the observed system $Ax = b + p$, where A, b, and p are defined in (40), by PLNA, LLNA and by least squares. These solutions are then evaluated by the observed system (training set) residual $\|Ax - b - p\|_1$ and the true system (testing set) residual $\|Ax - b\|_1$ and graphically comparing the recovered signal $\hat{g}(t)$ (37) to the true signal $g(t)$ (35).

In Figure 3(a) we plot the true signal, the observed signal and the signal recovered by solving, for one noise vector p, PLNA (25) with $\lambda = 0.30$ and LLNA (28) for $\lambda = 0.80$. Figure 3(b) displays the true signal, the observed signal and signal recovered for the same problem by least squares. The signal recovered by both PLNA and LLNA is considerably closer to the the true signal than that obtained by the least squares solution.

In this section we have proposed mathematical programming formulations which address two fundamental areas of predictive modeling: classification and regression. We now discuss a fundamentally different, very important data mining tool – clustering.

3 Clustering

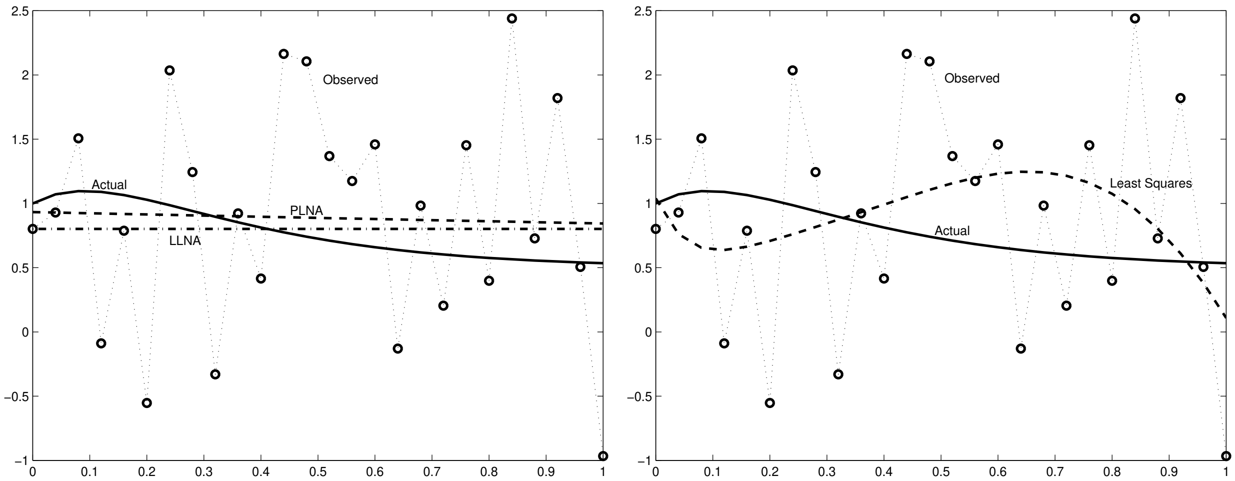
3.1 Overview

Given a finite sampling of points (or database) from a space $\mathcal{X}$, $\{x^i\}_{i=1}^M$, the target of of clustering or segmentation is to group the data into sets of "like" points. The goal being to obtain clusters which provide a high-level characterization of points belonging to an individual cluster. For instance, a cluster of similar objects may turn out to share a common cause or elements of a given cluster may relate to some important goal [110].

The fundamental difference between clustering and predictive modeling discussed previously is the classification function g. In the classification problem, we are given a sampling of this function over the training data (i.e. we "know" the class membership of the training data). In the clustering problem, we are attempting to *define* a "useful" classification function over the set $\{x^i\}_{i=1}^M$.

Unlike classification we usually do not know the number of desired "clusters" *a priori*. Clustering algorithms typically employ a two-stage search: An outer loop over possible cluster numbers and an inner loop to fit the best possible clustering for a given number of clusters. Given the number k of clusters, clustering methods can be divided into three classes:

1. Metric-distance based methods: a distance measure is defined and the objective becomes finding the best k-way partition such that cases in each block of the partition are closer to each other (or centroid) than to cases in other clusters.



(a) Dashed curves are the recovered signal $\hat{g}(t)$ with coefficient vector $x(\lambda)$ determined by (25) with $\lambda = 0.3$ and $\|Ax(\lambda) - b\|_1 = 4.7410$ for PLNA and by (28) with $\lambda = 0.8$ and $\|Ax(\lambda) - b\|_1 = 4.7076$ for LLNA.

(b) Dashed curve is the recovered signal $\hat{g}(t)$ with coefficient vector $x(ls)$ determined by **least squares** solution. Note: $\|Ax(ls) - b\|_1 = 8.9733$.

Figure 3: **Signal Recovery.** Solid curve is the true signal $g(t)$. Circles connected by dotted lines are the observed signal $\tilde{g}(t_i)$ sampled at discrete times and the dashed curves are the recovered signals.

2. Model-based methods: a model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each cluster. Let the data be denoted by $D := \{x^i\}_{i=1}^M$. If M_ℓ is the model hypothesized for cluster ℓ, ($\ell \in \{1,\dots,k\}$), then one way to score the fit of a model to a cluster is via the likelihood:

$$\text{Prob}(M_\ell|D) = \text{Prob}(D|M_\ell)\frac{\text{Prob}(M_\ell)}{\text{Prob}(D)}$$

   The prior probability of the data, Prob(D) is a constant and hence can be ignored for comparison purposes, while Prob(M_ℓ) is the prior assigned to a model. In maximum likelihood techniques, all models are assumed equally likely and hence this term is ignored. A problem with ignoring this term is that more complex models are always preferred and this leads to overfitting the data.

3. Partition-based methods: basically enumerate various partitions and then score them by some criterion. The above two techniques can be viewed as special cases of this class. Many techniques in the AI literature fall under this category and utilize ad hoc scoring functions.

In the next section we focus on mathematical programming formulations for *nonhierarchical clustering* in which the number of clusters or groups is fixed *a priori*. For a description of hierarchical clustering procedures, see [68].

3.2 Mathematical Programming Formulations

We first address the following explicit description of the clustering problem: given m points $\{x^1, x^2, \dots, x^m\}$ in n-dimensional real space R^n, and a fixed integer k of clusters, determine k "centers" in R^n, $\{c^1, c^2, \dots, c^k\}$, such that the sum of the "distances" of each point to a nearest cluster center is minimized. The clustering problem then is:

$$\min_{c^1,\dots,c^k} \sum_{i=1}^m \min_{\ell=1,\dots,k} \|x^i - c^\ell\|, \tag{41}$$

where the norm $\|\cdot\|$ is some arbitrary norm on R^n. Note that the objective function of (41) is the summation of the minimum of a set of convex functions which is, in general, neither convex nor concave. Hence this is a difficult optimization problem. We propose simplifying this problem slightly by introducing a "selection" variable $t_{i\ell}$ justified by the following simple lemma [23, Lemma 3.1].

Lemma 3.1 *Let $a \in R^k$. Then*

$$\min_{1\le\ell\le k}\{a_\ell\} = \min_{t_1,\dots,t_k}\left\{\sum_{\ell=1}^k t_\ell a_\ell \;\middle|\; \sum_{\ell=1}^k t_\ell = 1,\ t_\ell \ge 0, \ell = 1,\dots,k\right\}. \tag{42}$$

This simple lemma allows us to reformulate problem (41) as the following constrained optimization problem:

$$\begin{array}{ll} \underset{c^\ell, t_{i\ell}}{\text{minimize}} & \sum_{i=1}^m \sum_{\ell=1}^k t_{i\ell}\cdot\|x^i - c^\ell\| \\ \text{subject to} & \sum_{\ell=1}^k t_{i\ell} = 1, t_{i\ell} \ge 0,\ i = 1,\dots,m,\ \ell = 1,\dots,k \end{array} \tag{43}$$

Notice that for a fixed data point x^i, if $\bar\ell$ is the index such that center $c^{\bar\ell}$ is nearest to x^i, then $t_{i\bar\ell} = 1$ and $t_{i\ell} = 0$, $\ell \ne \bar\ell$. If multiple centers have the same minimum distance to x^i, then the t-values corresponding to these cluster centers will be nonzero and form a convex combination of this minimum distance.

We focus our attention on solving (43) with the 1-norm distance for which problem (43) can be written as the following bilinear program [23]:

$$\begin{array}{ll} \underset{c,d,t}{\text{minimize}} & \sum_{i=1}^m \sum_{\ell=1}^k t_{i\ell}\cdot(e^T d_{i\ell}) \\ \text{subject to} & -d_{i\ell} \le x^i - c^\ell \le d_{i\ell},\ i = 1,\dots,m,\ \ell = 1,\dots,k, \\ & \sum_{\ell=1}^k t_{i\ell} = 1,\ t_{i\ell} \ge 0,\ i = 1,\dots,m,\ \ell = 1,\dots,k. \end{array} \tag{44}$$

Here $d_{i\ell} \in R^n$ is a dummy variable that bounds the components of the difference $x^i - c^\ell$. Hence $e^T d_{i\ell}$ bounds the 1-norm distance between point x^i and center c^ℓ. By using the 1-norm, not only can the clustering problem (44) be formulated as a bilinear program, but the computed centers are less sensitive to outliers such as those resulting when the underlying data distributions have pronounced tails.

We further note that the constraints of (44) are uncoupled in the variables (c, d) and the variable t. Hence the Uncoupled Bilinear Programming Algorithm [10, Algorithm 2.1] is applicable. This algorithm alternates between solving a linear program in the variable t and a linear program in the variables (c, d) and terminates finitely at a point satisfying the minimum principle necessary optimality condition for problem (44) [10, Theorem 2.1]. Due to the simple structure of (44), the two linear programs can be solved explicitly in closed form. This leads to the following algorithm.

Algorithm 3.2 k-Median Algorithm. *Given k cluster centers $c^{1,j}, c^{2,j}, \ldots, c^{k,j}$ at iteration j, compute $c^{1,j+1}, c^{2,j+1}, \ldots, c^{k,j+1}$ by the following 2 steps:*

1. *Cluster Assignment: For $i = 1, \ldots, m$, assign x^i to cluster $l(i)$ such that $c^{l(i),j}$ is nearest to x^i in the 1-norm.*
2. *Cluster Center Update: For $\ell = 1, \ldots, k$ set $c^{\ell,j+1}$ to be the median of all x^i assigned to $c^{\ell,j}$.*
   Remark: The point $c^{\ell,j+1}$ is a cluster center that minimizes the sum of the 1-norm distances to all points in cluster ℓ.

Stop when $c^{\ell,j} = c^{\ell,j+1}$, $\ell = 1, \ldots, k$.

If we consider problem (43) with the 2-norm *squared*, the iterative algorithm is the popular k-Mean approach to clustering [68]. The underlying problem for the k-Mean algorithm is the following:

$$\begin{array}{ll} \underset{c,t}{\text{minimize}} & \sum_{\ell=1}^{k} \sum_{i=1}^{m} t_{i\ell} \left(\frac{1}{2} \|x^i - c^\ell\|_2^2 \right) \\ \text{subject to} & \sum_{\ell=1}^{k} t_{i\ell} = 1, \ t_{i\ell} \geq 0, \ i = 1, \ldots, m, \ \ell = 1, \ldots, k. \end{array} \tag{45}$$

The iterative approach essentially involves solving a linear program in t for a fixed value of c (cluster assignment step) and a quadratic program in c for a fixed value of t (cluster center update step). Again, these two problems can be solved in closed form.

Algorithm 3.3 k-Mean Algorithm. *Given k cluster centers $c^{1,j}, c^{2,j}, \ldots, c^{k,j}$ at iteration j, compute $c^{1,j+1}, c^{2,j+1}, \ldots, c^{k,j+1}$ by the following 2 steps:*

1. *Cluster Assignment: For each $i = 1, \ldots, m$, assign x^i to cluster $l(i)$ such that $c^{l(i),j}$ is nearest to x^i in the 2-norm.*
2. *Cluster Center Update: For $\ell = 1, \ldots, k$ set $c^{\ell,j+1}$ to be the mean of all x^i assigned to $c^{\ell,j}$.*
   Remark: The point $c^{\ell,j+1}$ is a cluster center that minimizes the sum of the 2-norm distances squared to all the points in cluster ℓ.

Stop when $c^{\ell,j} = c^{\ell,j+1}$, $\ell = 1, \ldots, k$.

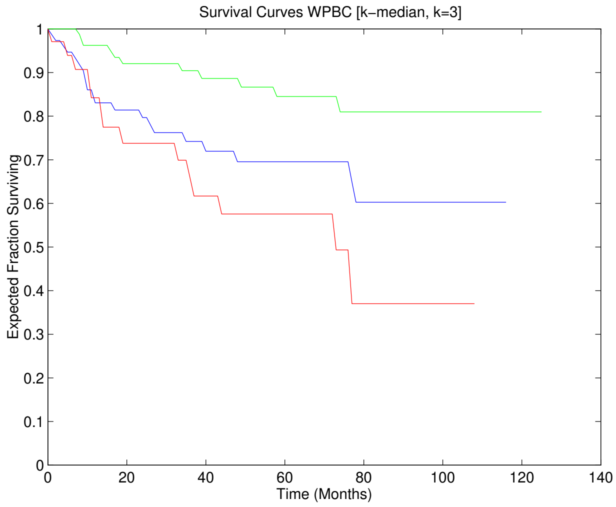
If the 2-norm (not 2-norm squared) is used in the objective of (43), the cluster center update subproblem becomes the considerably harder Weber problem [38, 100] which locates a center in R^n closest in sum of Euclidean distances to a finite set of given points. Choosing the mean of the points assigned to a given cluster as its center minimizes the sum of the 2-norm distances *squared* between the data points and the cluster center. This property makes the centers computed by the k-Mean algorithm less robust to outliers in the data.

A scalable k-Mean algorithm is discussed and evaluated in [17]. Scalable approaches to clustering also include the BIRCH algorithm [125], CURE [60] and CLIQUE [2].

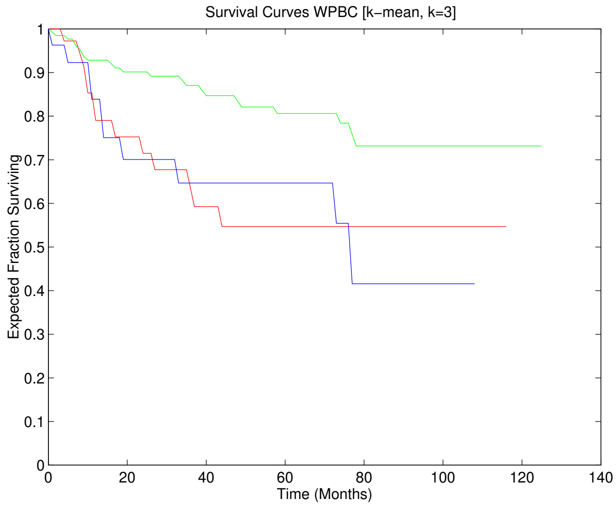
We did not address the problem of efficiently determining the initial placement of the k medians or means. This issue is specifically addressed in [16]. Determining the number of clusters k is a harder problem. Most practitioners run several clustering sessions with different values for k and choose the "best" result [34]. Cross-validation is a good method for choosing k [112].

We note that k-Mean convergence has been shown in [109] and a discussion relating to the convergence proof is presented in [4].

We next present results of applying these clustering techniques to extract clinically-important survival curves from a breast cancer prognosis database.



(a) k-Median



(b) k-Mean

Figure 4: Estimated fraction of disease free patients versus time (months) for 3 clusters obtained with the k-Median (a) and k-Mean Algorithms on the WPBC dataset.

3.3 Clustering Case Study

We present a case study comparing the k-Median and k-Mean algorithms. The k-Median and k-Mean algorithms were applied to two features of the Wisconsin Prognosis Breast Cancer dataset (available from UCI Repository of Machine Learning Databases, University of California, Irvine, http://www.ics.uci.edu/ ̃mlearn/MLRepository.html). The two features used were tumor size (diameter of the excised tumor in centimeters) and lymph node status (number of axillary lymph nodes metastasized at time of surgery). These two features were then normalized to have mean = 0 and standard deviation = 1. This dataset then consists of 194 points in R^2.

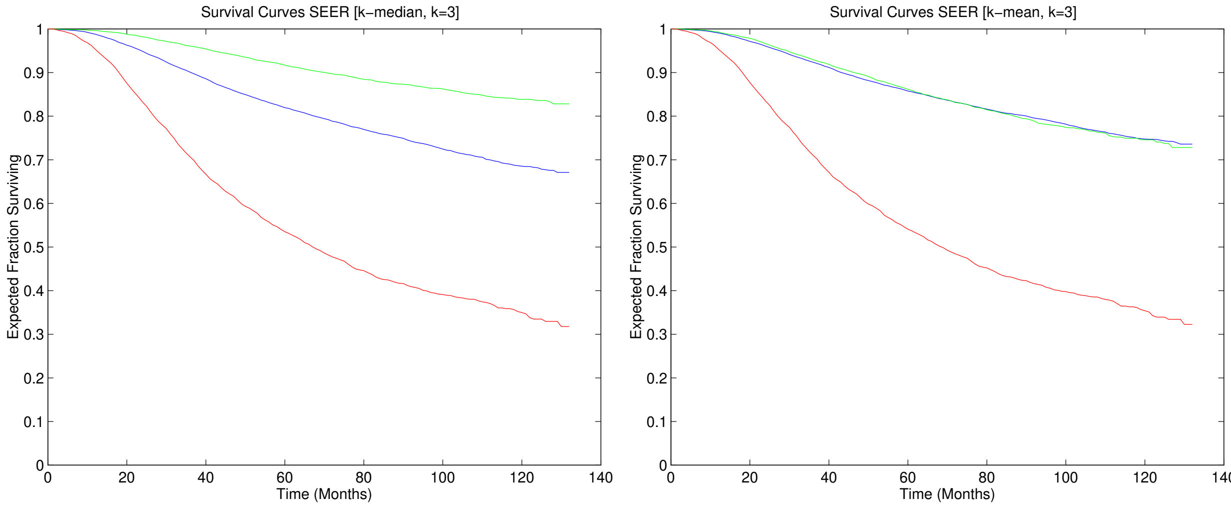
The k-Median and k-Mean algorithms were also applied to the SEER database [31] consisting of the two features of tumor size and nodes positive for 21,960 instances.

The k-Median and k-Mean Algorithms were applied to both datasets with $k = 3$. Survival curves [70] were then constructed for each cluster, representing expected percent of surviving patients as a function of time, for patients in that cluster. The value of $k = 3$ was chosen for the purpose of determining clusters that represented patients with "good", "average" and "poor" prognosis, as depicted by the survival curves. Initial cluster centers where chosen by first dividing the coordinate axes into nine intervals over the range of the data and choosing three centers as midpoints of the densest, second densest, and third densest intervals [23].

Figure 4(a) depicts survival curves computed from the WPBC dataset clustered by the k-Median Algorithm. Figure 4(b) depicts survival curves for clusters obtained with the k-Mean Algorithm applied to the WPBC dataset.

The key observation to make is that the curves in Figure 4(a) are *well separated*. Hence the clusters obtained for $k = 3$, by the k-Median Algorithm can be used as prognostic tools. In contrast, the curves in Figure 4(b) are poorly separated, and hence are not useful for prognosis.

Figure 5(a) depicts survival curves for three clusters obtained by the k-Median Algorithm on the SEER dataset. Again, since these survival curves are separated, the clusters can be used as prognostic indicators. In contrast, the two top curves in Figure 5(b), obtained by applying the k-Mean Algorithm to the SEER dataset, coalesce and hence are not useful for prognosis.



(a) k-Median

(b) k-Mean

Figure 5: Estimated fraction of disease free patients versus time (months) for 3 clusters obtained with the k-Median (a) and k-Mean Algorithms on the SEER dataset.

We note here that clustering can also be done via density estimation. Given k clusters, one can associate a probability density function (model) with each of the clusters. The problem then reduces to estimating the parameters of each model. This topic is included in the following discussion.

4 Dependency Modeling

4.1 Overview

Insight into data is often gained by deriving some causal structure within the data. Models of causality can be probabilistic (as in deriving some statement about the probability distribution governing the data) or they can be deterministic as in deriving functional dependencies between fields in the data [101]. Density estimation methods in general fall under this category, so do methods for explicit causal modeling (e.g. [58] and [63]). We focus our attention specifically on density estimation.

4.2 Density Estimation Mathematical Programming Formulations

In the density estimation problem [108], we are given a finite number of n-dimensional data points, that is $\{x^1, x^2, \ldots, x^M\}$. We assume that these data points are a finite sample from some unknown probability density function which maps a data point in R^n to the interval $[0, 1]$. The goal is to compute an estimate of the true probability density function. We will denote our *estimate* of the true probability density function (PDF) by $p(x)$. Once $p(x)$ is determined, then for any x in the domain of p, $p(x)$ is in the closed interval $[0, 1]$ indicating the probability of observing x. Parametric approaches to density estimation fix the functional form of the $p(x)$. The strength of the parametric approach comes from the ability to quickly compute $p(x)$ for any given x. In contrast nonparametric models allow very general forms of the estimate $p(x)$ but suffer in that the number of model variables grows directly with the number of data points [13]. We consider a semi-parametric approach to density estimation incorporating advantages of both the parametric and nonparametric models.

The semi-parametric approach considered here is the *mixture model*. Here, our estimate $p(x)$ of true PDF is a linear combination of k "basis functions", where k is a parameter of the model typically much less than the number of data points M. Aside from a slight change of notation, this section follows pages 60-67 of [13]. We estimate the PDF by:

$$p(x) = \sum_{\ell=1}^{k} p(x|\ell)P(\ell). \tag{46}$$

$P(\ell)$ is the *prior* probability of the data point having been generated by *component* ℓ of the mixture and $p(x|\ell)$ are the *conditional densities*. The "basis functions" are these conditional densities $p(x|\ell), \ell = 1, \dots, k$ and the linear coefficients are the prior probabilities $P(\ell), \ell = 1, \dots, k$. The prior probabilities satisfy:

$$\sum_{\ell=1}^{k} P(\ell) = 1, \; 0 \le P(\ell) \le 1; \ell = 1, \dots, k. \tag{47}$$

Similarly, the component functions $p(x|\ell)$ ("basis functions") are normalized so that

$$\int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} p(x|\ell)dx_1 dx_2 \dots dx_n = 1. \tag{48}$$

Once the estimate (46) is determined, it is a probability density function. To generate a data point with this PDF $p(x)$, one of the components ℓ is randomly selected with probability $P(\ell)$ and then a data point is generated from the corresponding density $p(x|\ell)$. $P(\ell)$ is the prior probability that a data point was generated from component ℓ of the mixture and $p(x|\ell)$ is the probability of the given data point under component ℓ.

We also introduce the notion of *posterior* probabilities, computed using Bayes' Theorem:

$$P(\ell|x) = \frac{p(x|\ell)P(\ell)}{p(x)}. \tag{49}$$

These posterior probabilities satisfy:

$$\sum_{\ell=1}^{k} P(\ell|x) = 1. \tag{50}$$

The value of $P(\ell|x)$ represents the probability that a particular component ℓ is responsible for generating data point x.

We now focus our attention on Gaussian mixture models where each of the k "basis functions" or components is chosen as a Gaussian distribution with mean $\mu^\ell \in R^n$ and a covariance matrix which is a scalar multiple of the identity, $\Sigma^\ell = (\sigma^\ell)^2 I \in R^{n \times n}, \; \ell = 1, \dots, k$. Then the component density functions ("basis functions") are given by:

$$p(x|\ell) = \frac{1}{(2\pi(\sigma^\ell)^2)^{\frac{n}{2}}} \exp\left(\frac{-\|x - \mu^\ell\|_2^2}{2(\sigma^\ell)^2}\right). \tag{51}$$

For Gaussian components of this form, the estimate (46) has as adjustable parameters $P(\ell), \mu^\ell \in R^n$ and $\sigma^\ell \in R, \; \ell = 1, \dots, k$. The problem then reduces to estimating these parameters from the given data sample $\{x^1, \dots, x^M\} \subset R^n$.

The maximum likelihood approach determines values of these parameters that maximizes the likelihood that the observed data $\{x^1, \dots, x^M\}$ was actually sampled from the estimated PDF $p(x)$ (46). These values are found by minimizing the negative log-likelihood for the given data:

$$\min - \sum_{i=1}^{M} \log \left(\sum_{\ell=1}^{k} p(x^i|\ell) P(\ell) \right). \tag{52}$$

Minimizing the negative log-likelihood for the case of the Gaussian mixture model then becomes:

$$\min_{P(\ell),\mu^\ell,\sigma^\ell} - \sum_{i=1}^{M} \left(\log \left(\sum_{\ell=1}^{k} \frac{1}{(2\pi(\sigma^\ell)^2)^{\frac{n}{2}}} \exp \left(\frac{-\|x^i - \mu^\ell\|_2^2}{2(\sigma^\ell)^2} \right) P(\ell) \right) \right). \tag{53}$$

Problem (53) is a nontrivial optimization problem which has many local minima. The Expectation-Maximization (EM) algorithm [40] iteratively minimizes (53) for $P(\ell),\ \ell = 1, \dots, k$ for fixed values of μ^ℓ and σ^ℓ, then minimizes (53) for $\mu^\ell, \sigma^\ell, \ell = 1, \dots, k$ for fixed $P(\ell)$. The EM algorithm has been shown to converge to a local minima of (53) [97]. We now summarize the EM algorithm for the Gaussian mixture model.

Algorithm 4.1 Expectation-Maximization (EM) Algorithm. *Given $P^j(\ell) \in R, \mu^{j,\ell} \in R^n, \sigma^{j,\ell} \in R, \ell = 1, \dots, k$ at iteration j, compute $P^{j+1}(\ell), \mu^{j+1,\ell}, \sigma^{j+1,\ell}$ at iteration $j+1$ in the following 3 steps:*

1. *Posterior Calculation: Set*

$$P^j(\ell|x^i) = \frac{p^j(x^i|\ell)P^j(\ell)}{\sum_{\ell=1}^{k} p^j(x^i|\ell)P^j(\ell)},\ i = 1, \dots, m, \ell = 1, \dots, k, \tag{54}$$

where

$$p^j(x^i|\ell) = \frac{1}{(2\pi(\sigma^{j,\ell})^2)^{\frac{n}{2}}} \exp \left(\frac{-\|x^i - \mu^{j,\ell}\|_2^2}{2(\sigma^{j,\ell})^2} \right). \tag{55}$$

2. *Gaussian Parameter Update: for $\ell = 1, \dots, k$ set*

$$\mu^{j+1,\ell} = \frac{\sum_{i=1}^{M} P^j(\ell|x^i)x^i}{\sum_{i=1}^{M} P^j(\ell|x^i)}, \tag{56}$$

$$(\sigma^{j+1,\ell})^2 = \frac{1}{n} \frac{\sum_{i=1}^{M} P^j(\ell|x^i)\|x^i - \mu^{j+1,\ell}\|_2^2}{\sum_{i=1}^{M} P^j(\ell|x^i)}. \tag{57}$$

3. *Prior Probability Update: for $\ell = 1, \dots, k$ set*

$$P^{j+1}(\ell) = \frac{1}{M} \sum_{i=1}^{M} P^j(\ell|x^i). \tag{58}$$

Stop when $\mu^{j+1,\ell} = \mu^{j,\ell}, \sigma^{j+1,\ell} = \sigma^{j,\ell}$ and $P^{j+1,\ell} = P^{j,\ell},\ \ell = 1, \dots, k$.

We note that the EM algorithm can effectively be used to model databases with discrete attributes (dimensions) as well as continuous ones by assuming appropriate probability distributions for each of the k model components, $p(x|\ell),\ \ell = 1, \ldots, k$ (e.g. a Gaussian distribution over continuous dimensions and multinomial distributions over discrete dimensions). The EM approach to modeling is very general and flexible.

The relationship between the hard assignment method of the k-Mean algorithm (Algorithm 3.3) and the EM Algorithm can be found in [71]. In [97] a more general treatment of EM and the underlying optimization problem can be found.

A scalable EM algorithm is discussed and evaluated in [18].

The issue of choosing initial parameter values for EM is discussed in [46]. Choosing the value of k is difficult for EM as well as k-Median and k-Mean.Methods include trying several different values [34] or choosing k via cross-validation [112].

In the following section we provide an overview of the data mining methods of data summarization and change and deviation detection.

5 Other Methods

5.1 Data Summarization

Sometimes the goal of a data mining method is to simply extract compact patterns that describe subsets of the data. There are two classes of methods which represent taking horizontal (cases) or vertical (fields) slices of the data. In the former, one would like to produce summaries of subsets: e.g. producing sufficient statistics, or logical conditions that hold for subsets. In the latter case, one would like to predict relations between fields. This class of methods is distinguished from the other data mining methods discussed in that rather than predicting a specified field (e.g. classification) or grouping cases together (e.g. clustering) the goal is to find relations between fields. One common method is by *association rules* [3]. Associations are rules that state that certain combinations of values occur with other combinations of values with a certain frequency and certainty. A common application of this is market basket analysis were one would like to summarize which products are bought with what other products. While there are exponentially many rules, due to data sparseness only few such rules satisfy given support and confidence thresholds. Scalable algorithms find all such rules in linear time (for reasonable threshold settings). While these rules should not be viewed as statements about causal effects in the data, they are useful for modeling purposes if viewed as frequent marginals in a discrete (e.g. multinomial) probability distribution. Of course to do proper inference one needs to know the frequent, infrequent, and all probabilities in between. However, approximate inference can sometimes be useful.

5.2 Change and Deviation Detection

These methods account for sequence information, be it time-series or some other ordering (e.g. protein sequencing in genome mapping). The distinguishing feature of this class of methods is that ordering of observations is important and must be accounted for. Scalable methods for finding frequent sequences in databases, while in the worst-case exponential in complexity, do appear to execute efficiently given sparseness in real-world transactional databases [93].

6 Research Challenges

Successful KDD applications continue to appear, driven mainly by a glut in databases that have clearly grown to surpass raw human processing abilities. For examples of success stories in applications in industry see [15] and in science analysis see [43]. More detailed case studies are found in [45]. Driving the growth of this field are strong forces (both economic and social) that are a product of the data overload phenomenon. We view the need to deliver workable solutions to pressing problems as a very healthy pressure on the KDD field. Not only will it ensure our growth as a new engineering discipline, but it will provide our efforts with a healthy dose of reality checks; insuring that any theory or model that emerges will find its immediate real-world test environment.

The fundamental problems are still as difficult as they have been for the past few centuries as people considered difficulties of data analysis and how to mechanize it.

We first list the general challenges in KDD and data mining, and then we consider particular challenges for mathematical programming approaches to data mining problems.

6.1 General Research Challenges

The challenges facing advances in this field are formidable. Some of these challenges include:

1. Develop mining algorithms for classification, clustering, dependency analysis, summarization, and change and deviation detection that scale to large databases. There is a tradeoff between performance and accuracy as one surrenders to the fact that data resides primarily on disk or on a server and cannot fit in main memory. The majority of approaches that assume that data can fit in main memory need to be revised or redesigned.

2. Develop sampling approaches that are tightly coupled with mining algorithms and database access methods. Selecting a random sample from a database can be as expensive as scanning the entire data. In addition, any type of conditioning or stratification of the samples requires expensive indexing structures. Since mining algorithms typically attempt a multitude of conditioning situations, specialized random sampling schemes that exploit this behavior need to be developed.

3. Develop schemes for encoding "metadata" (information about the content and meaning of data) over data tables so that mining algorithms can operate meaningfully on a database and so that the KDD system can effectively ask for more information from the user.

4. While operating in a very large sample size environment is a blessing against overfitting problems, data mining systems need to guard against fitting models to data by chance. This problem becomes significant as a program explores a huge search space over many models for a given data set. As one enumerates many models, the probability that one of them will fit the data at hand by random chance approaches one. Proper adjustments and tests against holdout sets are crucial.

5. Develop effective means for data sampling, data reduction, and dimensionality reduction that operate on a mixture of categorical and numeric data fields. While large sample sizes allow us to handle higher dimensions, our understanding of high dimensional spaces and estimation within them is still fairly primitive. The curse of dimensionality is still with us.

6. Develop schemes capable of mining over nonhomogeneous data sets (including mixtures of multimedia, video, and text modalities) and deal with sparse relations that are only defined over parts of the data.

7. Develop new mining and search algorithms capable of extracting more complex relationships between fields and able to account for structure over the fields (e.g. hierarchies, sparse relations); i.e. go beyond the flat file or the single table assumption.

8. Develop data mining methods that account for prior knowledge of data and exploit such knowledge in reducing search, that can account for costs and benefits, and that are robust against uncertainty and missing data problems.

9. Enhance database management systems to support new primitives for the efficient extraction of necessary sufficient statistics as well as more efficient sampling schemes. This includes providing SQL support for new primitives that may be needed (e.g. [59]). Sufficient statistics are properties of the data that, from the perspective of the mining algorithm, eliminate the need for the data. Examples of sufficient statistics include histograms, counts, and sometimes data samples.

10. Scale methods to parallel databases with hundreds of tables, thousands of fields, and terabytes of data. Issues of query optimization in these settings are fundamental.

11. Account for and model comprehensibility of extracted models; allow proper tradeoffs between complexity and understandability of models for purposes of visualization and reporting; enable interactive exploration where the analyst can easily provide hints to help the mining algorithm with its search.

12. Develop theory and techniques to model growth and change in data. Large databases, because they grow over a long time, do not typically grow as if sampled from a static joint probability density. The question of how does the data grow needs to be better understood and tools for coping with it need to be developed. (See articles by P. Huber, by Fayyad & Smyth, and by others in [72])

13. Develop a theory and techniques for assessing significance in the presence of the large. Traditional techniques for assessing statistical significance were designed to handle the small sample case: in the presence of large data sets, these measures lose their intended "filtering" power.

14. Incorporate basic data mining methods with "any-time" analysis that can quantify and track the trade-off between accuracy and available resources, and optimize algorithms accordingly. Any-time algorithms are capable of providing "best solution so far" during their determination of the final solution. This is a desirable property of algorithms that need to run for extended periods of time.

6.2 Challenges for Mathematical Programming

We list below some of the issues that are of particular relevance to data mining and potential mathematical programming approaches for their resolution.

1. Scaling to large sets of constraints and variables.

   (a) Decomposing constraints and variables into subsets. The parallel constraint distribution approach of [49] is one way of decomposing problem constraints among parallel processors or parallel virtual machines (PVM) [56, 77], while the parallel variable distribution of [50] can be similarly applied when dealing with databases with a very large number of attributes.

   (b) Algorithms for dealing with constraints and variables sequentially. Mathematical programming approaches that deal with problem data in a sequential manner are very useful for processing large databases. Such incremental approaches have already been developed for establishing convergence of the online back-propagation algorithm that trains neural networks incrementally [90, 89] as well as for handling more general types of problems [118, 113].

2. Approximate solutions.

   (a) The problem of identifying constraints which are inactive and will remain inactive as optimization proceeds is an important and interesting practical problem. Computational algorithms towards this goal have appeared in [30, 52, 124, 12]. A possible simple approach for this problem is the use of the classical exterior quadratic penalty function [51] and dropping constraints for which the product of the penalty parameter (which is increasing to infinity) times the constraint violation is less than some tolerance. This product approximates the Lagrange multiplier for the constraint. A small or zero value of this product is indicative of a small multiplier and hence an inactive constraint.

3. Mathematical programming languages geared for data mining problems.
   Although there are languages specifically designed for setting up and solving mathematical programs, such as AMPL [53] and GAMS [27, 67], and languages that can be easily adapted for the same purpose, such as MATLAB [94], no specific language has been designed for mathematical programming applications to data mining. Such a language could make such applications more widely accepted by the data mining community.

4. Data reduction techniques.
   Removing redundant or irrelevant data from a mathematical programming application to data mining is one of the key requirements for handling large databases and extracting knowledge from them via a generalization paradigm. As such we believe that the use of the step function $(\cdot)_*$ and its surrogate approximation, the exponential of equation (15), as an objective function penalty on the problem variables, is one of the most effective ways for simplifying the problem and reducing data size. This approach, which

leads to minimization of a concave function on (often) a polyhedral set [85], has been used very effectively for feature selection [24] and noise suppression [22]. Although the polyhedral concave minimization problem is a difficult problem, fast and finite algorithms for finding very useful stationary points make it a very effective tool for data reduction. Further study of the problem and other approaches constitute important research areas.

5. Effect of data perturbation on derived models.
   Perturbation of mathematical programs is a widely studied area, for example [105, 104, 5, 48, 76]. Applying these results to specific data mining problems where the data is constantly being augmented and revised without having to rebuild the mathematical programming model would be a valuable contribution.

6. Modeling noise.
   Mathematical programming models that purposely tolerate error, either because there is noise in the data or because the model is an inaccurate representation of the real problem, are likely to perform better. One such approach, tolerant training [115], purposely tolerates such inaccuracies in the model, and often leads to better predictive results. Extending this tolerant mathematical programming model to a wider class of problems would be an important and practical contribution.

7. Visualization and understandability of derived models.
   Presenting a model visually enhances its utility and increases the chances of its correct usage. Many mathematical programming models possess this property. For example the robust linear separator of [9] has a very simple geometric representation as a plane separating most or all the elements of two sets, which has been used to advantage in medical applications [92, 91]. Similarly the multisurface separation method [81, 88] can be geometrically depicted as placing the sets to be separated into distinct polyhedral compartments.

8. Local versus global methods.
   Because of size considerations we often do not want to model the whole data set. We therefore need to pre-partition the data and model it locally. One way to achieve this is via a decision tree approach [6, 96] in which the data is partitioned into local regions and modeled individually. A regression approach to this is given by Regression trees in CART [26]. Another example is to cluster the data first, then model each cluster locally.

9. Modeling rare events (e.g. low probability items like fraud or rare diseases).
   This is an important problem which can be possibly addressed by a careful use of an appropriate norm in the context of a separation problem. For example the use of the infinity norm as originally proposed in [80, 81] can be used for capturing such rare events. In general, objects that occurr with a very low frequency (e.g. in fraud detection applications or in detecting quasars in sky surveys in astronomy [47]) are likely to be dismissed as insignificant outliers or simply disappear in L2-norm based methods and standard principal component analysis.

10. Going beyond L-norms.
    Current approaches utilize all dimensions of a problem with equal weight or use feature selection to weight each dimension with a 0 or 1 weight. Allowing a variable weight in the interval $[0, 1]$ for example would introduce a scaling of the problem that could enhance separation either by a linear or nonlinear separating approach [80, 81, 9].

7 Concluding Remarks

KDD holds the promise of an enabling technology that could unlock the knowledge lying dormant in huge databases. Perhaps the most exciting aspect is the possibility of the evolution of a new breed of methods properly mixing statistics, databases, optimization, automated data analysis and reduction, and other related areas. This mix would produce new algorithms and methodologies, tuned to working on large databases and scalable both in data set size and in parallelism.

In this paper, we provided an overview of this area, defined the basic terms, and covered some of the basic research challenges. We included mathematical programming formulations of some data mining problems and provided sample case studies of mathematical programming approaches to data mining applications. We also outlined prospects and challenges for the role of mathematical programming in KDD. While KDD will draw on the substantial body of knowledge built up in its constituent fields, it is inevitable that a new science will emerge as these challenges are addressed, and that suitable mixtures of ideas from each of the constituent disciplines will greatly enhance our ability to exploit massive, ever-growing, and ever-changing data sets.

Acknowledgements

O. L. Mangasarian is supported by National Science Foundation Grant CCR-9322479 and Air Force Office of Scientific Research Grant F49620-97-1-0326.

References

[1] T. J. Abatzoglou, J. M. Mendel, and G. A. Harada. The constrained total least squares technique and its application to harmonic superposition. *IEEE Transactions on Signal Processing*, 39:1070–1087, 1991.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applicaitons. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 94–105, New York, 1998. ACM.

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and I.C. Verkamo. Fast discovery of association rules. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in knowledge Discovery and Data Mining*, pages 307 – 328. MIT Press, Cambridge, MA, 1996.

[4] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, 1973.

[5] B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer. *Nonlinear Parametric Optimization.* Akamie-Verlag, Berlin, 1982.

[6] K. P. Bennett. Decision tree construction via linear programming. In M. Evans, editor, *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference*, pages 97–101, Utica, Illinois, 1992.

[7] K. P. Bennett and J. A. Blue. A support vector machine approach to decision trees. In *Proceedings of IJCNN'98*, pages 2396–2401, Anchorage, Alaska, 1997. http://www.math.rpi.edu/∼bennek/.

[8] 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.

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

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

[11] K. P. Bennett and O. L. Mangasarian. Multicategory separation via linear programming. *Optimization Methods and Software*, 3:27–39, 1993.

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

[13] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[14] S. Bös. A realizable learning task which shows overfitting. In *Advances in Neural Information Processing Systems 8*, pages 218–224, Cambridge, MA, 1996. The MIT Press.

[15] R. Brachman, T. Khabaza, W. Kloesgen, G. Piatetsky-Shapiro, and E. Simoudis. Industrial applications of data mining and knowledge discovery. *Communications of ACM*, 39(11), 1996.

[16] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*, pages 91–99, San Francisco, CA, 1998. Morgan Kaufmann. http://www.research.microsoft.com/research/dtg/fayyad/papers/icml98.htm.

[17] P. S. Bradley, U. M. Fayyad, and C. Reina. Scaling clustering to large databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, KDD98*, pages 9–15, Menlo Park, CA, 1998. AAAI Press. http://www.research.microsoft.com/research/dtg/fayyad/papers/kdd98-cluster.html.

[18] P. S. Bradley, U. M. Fayyad, and C. Reina. Scaling EM clustering to large databases. Technical Report MSR-TR-98-35, Microsoft Research, Redmond, WA, 1998. 1999 ACM SIGMOD International Conference on Management of Data, submitted.

[19] P. S. Bradley and O. L. Mangasarian. Parsimonious side propagation. Technical Report 97-11, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, October 1997. ICASSP98: IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle May 12-15, 1998, Volume 3, pages 1873-1876. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-11.ps.Z.

[20] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference(ICML '98)*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.Z.

[21] P. S. Bradley and O. L. Mangasarian. Massive data discrimination via linear support vector machines. Technical Report 98-05, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, May 1998. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.Z.

[22] P. S. Bradley, O. L. Mangasarian, and J. B. Rosen. Parsimonious least norm approximation. *Computational Optimization and Applications*, 11(1):5–21, October 1998. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-03.ps.Z.

[23] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems -9-*, pages 368–374, Cambridge, MA, 1997. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/96-03.ps.Z.

[24] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-21.ps.Z.

[25] E. J. Bredensteiner and K. P. Bennett. Feature minimization within decision trees. *Computational Optimizations and Applications*, 10:111–126, 1998.

[26] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth & Brooks, Monterey, CA, 1984.

[27] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A User's Guide.* The Scientific Press, South San Francisco, CA, 1988.

[28] C. J. C. Burges. Simplified support vector decision rules. In L. Saita, editor, *Machine Learning–Proceedings of the Thirteenth International Conference (ICML '96)–Bari, Italy July 3-6, 1996*, pages 71–77, San Francisco, CA, 1996. Morgan Kaufmann.

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

[30] J. V. Burke and J. J. Moré. On the identification of active constraints. *SIAM Journal of Numerical Analysis*, 25(5):1197–1211, 1988.

[31] C. L. Carter, C. Allen, and D. E. Henson. Relation of tumor size, lymph node status, and survival in 24,740 breast cancer cases. *Cancer*, 63:181–187, 1989. SEER: Surveillence, Epidemiology and End Results Program of the National Cancer Institute.

[32] S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. In *ACM SIGMOD RECORD*, volume March, 1997.

[33] S. Chaudhuri, U. M. Fayyad, and J. Bernhardt. Scalable classification over sql databases. Technical Report MSR-TR-97-35, Microsoft Research, Redmond, WA, 1997.

[34] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 153–180, Menlo Park, CA, 1996. AAAI Press.

[35] Chunhui Chen and O. L. Mangasarian. Hybrid misclassification minimization. *Advances in Computational Mathematics*, 5(2):127–136, 1996. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-05.ps.Z.

[36] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, Stanford, California 94305, February 1996. Available by http://playfair.stanford.EDU/reports/chen_s/BasisPursuit.ps.Z.

[37] E.F. Codd. Providing olap (on-line analytical processing) to user-analysts: An it mandate. Technical report, E.F. Codd and Associates, 1993.

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

[39] I. Daubechies. Time-frequency localization operators: A geometric phase space approach. *IEEE Transactions on Information Theory*, 34:605–612, 1988.

[40] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[41] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973.

[42] U. Fayyad and R. Uthurusamy (Eds.). Special issue on data mining. *Communications of The ACM*, vol. 39, no. 11 1996.

[43] U. Fayyad, D. Haussler, and P. Stolorz. Mining science data. *Communications of ACM*, 39(11), 1996.

[44] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in knowledge Discovery and Data Mining*, pages 1 – 36. MIT Press, Cambridge, MA, 1996.

[45] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, MA, 1996.

[46] U. M. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, KDD98*, pages 194–198, Menlo Park, CA, 1998. AAAI Press. http://www.research.microsoft.com/research/dtg/fayyad/papers/kdd98-init.html.

[47] U.M. Fayyad, S.G. Djorgovski, and N. Weir. Application of classification and clustering to sky survey cataloging and analysis. In E. Wegman and S. Azen, editors, *Computing Science and Statistics*, volume 29(2), pages 178 – 186, Fairfax, VA, USA, 1997. Interface Foundation of North America.

[48] M. C. Ferris and O. L. Mangasarian. Finite perturbation of convex programs. *Applied Mathematics and Optimization*, 23:263–273, 1991.

[49] M. C. Ferris and O. L. Mangasarian. Parallel constraint distribution. *SIAM Journal on Optimization*, 1(4):487–500, 1991.

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

[51] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, New York, 1968.

[52] S. D. Flam. On finite convergence and constraint identification of subgradient projection methods. *Mathematical Programming*, 57:427–437, 1992.

[53] R. Fourer, D. Gay, and B. Kernighan. *AMPL*. The Scientific Press, South San Francisco, California, 1993.

[54] J. Friedman. On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1), 1997.

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

[56] A. Geist, A. Beguelin, Dongarra J, W. Jiang, R. Mancheck, and V. Sunderam. *PVM - Parallel Virtual Machine*. MIT Press, Cambridge, MA, 1994.

[57] A. A. Giordano. *Least Square Estimation With Applications to Digital Signal Processing*. John Wiley & Sons, New York, 1985.

[58] C. Glymour, R. Scheines, and P. Spirtes ABD K. Kelly. *Discovering Causal Structure*. Academic Press, New York, 1987.

[59] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. *Data Mining and Knowledge Discovery*, 1(1), 1997.

[60] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, New York, 1998. ACM.

[61] V. Harinarayan, A. Rajaraman, and J. Ullman. Implementing data cubes efficiently. In *Proceeding of the ACM SIGMOD-96 Conference*, 1996.

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

[63] D. Heckerman. Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1(1), 1997.

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

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

[66] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem, Computational Aspects and Analysis*. SIAM, Philadelphia, PA, 1991.

[67] IBM Optimization Subroutine Library. *GAMS- The Solver Manuals: OSL*. GAMS Development Corporation, Washington, D.C., 1994.

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

[69] 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*, pages 121–129, San Mateo, CA, 1994. Morgan Kaufmann.

[70] E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53:457–481, 1958.

[71] M. Kearns, Y. Mansour, and A. Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 282–293. Morgan Kaufmann, 1997.

[72] J. Kettenring and D. Pregibon, editors. *Statistics and Massive Data Sets, Report to the Committee on Applied and Theoretical Statistics*, Washington, D.C., 1996. National Research Council.

[73] K. Koller and M. Sahami. Toward optimal feature selection. In *Machine Learning: Proceedings of the Thirteenth International Conference*, San Mateo, CA, 1996. Morgan Kaufmann.

[74] 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.

[75] Edward E. Leamer. *Specification searches: ad hoc inference with nonexperimental data*. Wiley, New York, 1978.

[76] W. Li. Sharp condition constants for feasible and optimal solutions of a perturbed linear program. *Linear Algebra and Its Applications*, 187:15–40, 1993.

[77] M. Litzkow and M. Livny. Experience with the condor distributed batch system. In *Proceedings of the IEEE Workshop on Experimental Distributed Systems*, pages 97–101, Hunstville, AL, October 1990. IEEE Compter Society Press.

[78] M. Livny, R. Ramakrishnan, and T. Zhang. Fast density and probability estimations using CF-kernal method for very large databases. Technical report, University of Wisconsin, July 1996.

[79] Z.-Q. Luo, J.-S. Pang, and D. Ralph. *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press, Cambridge, England, 1996.

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

[81] O. L. Mangasarian. Multi-surface method of pattern separation. *IEEE Transactions on Information Theory*, IT-14:801–807, 1968.

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

[83] O. L. Mangasarian. Misclassification minimization. *Journal of Global Optimization*, 5:309–323, 1994.

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

[85] 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. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-20.ps.Z.

[86] 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.

[87] O. L. Mangasarian. Arbitrary-norm separating plane. Technical Report 97-07, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, May 1997. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-07.ps.Z.

[88] O. L. Mangasarian, R. Setiono, and W. H. Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis. In T. F. Coleman and Y. Li, editors, *Large-Scale Numerical Optimization*, pages 22–31, Philadelphia, Pennsylvania, 1990. SIAM. Proceedings of the Workshop on Large-Scale Numerical Optimization, Cornell University, Ithaca, New York, October 19-20, 1989.

[89] O. L. Mangasarian and M. V. Solodov. Backpropagation convergence via deterministic nonmonotone perturbed minimization. In G. Tesauro J. D. Cowan and J. Alspector, editors, *Advances in Neural Information Processing Systems -6-*, pages 383–390, San Francisco, CA, 1994. Morgan Kaufmann.

[90] 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.

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

[92] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23:1 & 18, 1990.

[93] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovery of frequent episodes in event sequence. *Data Mining and Knowledge Discovery*, 1(3), 1997.

[94] MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1992.

[95] M. Mehta, R. Agrawal, and J. Rissanen. Sliq: a fast scalable classifier for data mining. In *Proceedings of EDBT-96*. Springer Verlag, 1996.

[96] S. Murthy, S. Kasif, S. Salzberg, and R. Beigel. OC1: Randomized induction of oblique decision trees. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 322–327, Cambridge, MA 02142, 1993. The AAAI Press/The MIT Press.

[97] R. M. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. Technical report, Dept. of Statistics and Dept. of Computer Science, University of Toronto, Toronto, Ontario, Canada, 1997.

[98] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. In *Proceedings of IEEE NNSP'97, Amelia Island, FL, September 1997, 276-285*, 1997. http://www.ai.mit.edu/people/girosi/home-page/svm.html.

[99] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, June 1997, 130-136*, 1997. http://www.ai.mit.edu/people/girosi/home-page/svm.html.

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

[101] G. Piatetsky-Shapiro and W. Frawley, editors. *Knowledge Discovery in Databases*. MIT Press, Cambridge, MA, 1991.

[102] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1999. http://www.research.microsoft.com/~jplatt/smo.html.

[103] J. Rissanen. Modeling by shortest data description. *Automatica*, pages 465–471, 1978.

[104] S. M. Robinson. Bounds for error in the solution of a perturbed linear program. *Linear Algebra and Its Applications*, 6:69–81, 1973.

[105] S. M. Robinson. Perturbations in finite-dimensional systems of linear inequalities and equations. Technical Summary Report No. 1357, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, August 1973.

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

[107] B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *Proceedings of the First International Conference in Knowledge Disovery and Data Mining*, Menlo Park, CA, 1995. AAAI Press.

[108] D. W. Scott. *Multivariate Density Estimation.* John Wiley and Sons, New York, 1992.

[109] 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.

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

[111] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In U. Fayyad and R. Uthurusamy, editors, *Proceedings of KDD-95: First International Conference on Knowledge Discovery and Data Mining*, pages 275–281, Menlo Park, CA, 1995. AAAI Press.

[112] P. Smyth. Clustering using monte carlo cross-validation. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD96)*, Menlo Park, CA, 1996. AAAI Press.

[113] M. V. Solodov. Incremental gradient algorithms with stepsizes bounded away from zero. *Computational Optimization and Applications*, 14(1), 1999. To appear.

[114] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, 36:111–147, 1974.

[115] W. N. Street and O. L. Mangasarian. Improved generalization via tolerant training. *Journal of Optimization Theory and Applications*, 96(2):259–279, February 1998. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-11.ps.Z.

[116] K. Sung and T. Poggio. Example-based learning for view-based human face detection. A. I. Lab, A. I. Memo 1521, MIT, Cambridge, MA, December 1994.

[117] C. W. Therrien. *Discrete Random Signals and Statistical Signal Processing.* Prentice-Hall, Inc, Englewood Cliffs, NJ, 1992.

[118] P. Tseng. Incremental Gradient(-Projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8, 1998. To appear.

[119] V. Vapnik, S. E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, Cambridge, MA, 1997. MIT Press.

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

[121] G. Wahba. *Spline Models for Observational Data.* SIAM, Philadelphia, 1990.

[122] G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 69–88, Cambridge, MA, 1999. MIT Press. ftp://ftp.stat.wisc.edu/pub/wahba/index.html.

[123] C.S. Wallace and J.D. Patrick. Coding decision trees. Technical Report TR 151, Monash University, Melbourne, Australia, 1991.

[124] S. J. Wright. Identifiable surfaces in constrained optimization. *SIAM Journal on Control and Optimization*, 31:1063–1079, 1993.

[125] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.