Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research xxx (2006) xxx–xxx
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Operations research and data mining

Sigurdur Olafsson [*], Xiaonan Li, Shuning Wu

Department of Industrial and Manufacturing Systems Engineering, Iowa State University, 2019 Black Engineering, Ames, IA 50011, USA

Abstract

With the rapid growth of databases in many modern enterprises data mining has become an increasingly important approach for data analysis. The operations research community has contributed significantly to this field, especially through the formulation and solution of numerous data mining problems as optimization problems, and several operations research applications can also be addressed using data mining methods. This paper provides a survey of the intersection of operations research and data mining. The primary goals of the paper are to illustrate the range of interactions between the two fields, present some detailed examples of important research work, and provide comprehensive references to other important work in the area. The paper thus looks at both the different optimization methods that can be used for data mining, as well as the data mining process itself and how operations research methods can be used in almost every step of this process. Promising directions for future research are also identified throughout the paper. Finally, the paper looks at some applications related to the area of management of electronic services, namely customer relationship management and personalization.

Keywords: Data mining; Optimization; Classification; Clustering; Mathematical programming; Heuristics

1. Introduction

In recent years, the field of data mining has seen an explosion of interest from both academia and industry. Driving this interest is the fact that data collection and storage has become easier and less expensive, so databases in modern enterprises are now often massive. This is particularly true in web-based systems and it is therefore not surprising that data mining has been found particularly useful in areas related to electronic services. These massive databases often contain a wealth of important data that traditional methods of analysis fail to transform into relevant knowledge. Specifically, meaningful knowledge is often hidden and unexpected, and hypothesis driven methods, such as on-line analytical processing (OLAP) and most statistical methods, will generally fail to uncover such knowledge. Inductive methods, which learn directly from the data without an a priori hypothesis, must therefore be used to uncover hidden patterns and knowledge.

We use the term data mining to refer to all aspects of an automated or semi-automated process for extracting previously unknown and potentially useful knowledge and patterns from large databases. This process consists of numerous steps such as integration of data from numerous databases,

[*] Corresponding author. Tel.: +1 515 294 8908; fax: +1 515 294 3524.
E-mail address: olafsson@iastate.edu (S. Olafsson).

0377-2217/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2006.09.023

preprocessing of the data, and induction of a model with a learning algorithm. The model is then used to identify and implement actions to take within the enterprise. Data mining traditionally draws heavily on both statistics and machine learning but numerous problems in data mining can also be formulated as optimization problems (Freed and Glover, 1986; Mangasarian, 1997; Bradley et al., 1999; Padmanabhan and Tuzhilin, 2003).

All data mining starts with a set of data called the training set that consists of instances describing the observed values of certain variables or attributes. These instances are then used to learn a given target concept or pattern and, depending upon the nature of this concept, different inductive learning algorithms are applied. The most common concepts learned in data mining are classification, data clustering, and association rule discovery, and of those will be discussed in detail in Section 3. In classification the training data is labeled, that is, each instance is identified as belonging to one of two or more classes, and an inductive learning algorithm is used to create a model that discriminates between those class values. The model can then be used to classify any new instances according to this class attribute. The primary objective is usually for the classification to be as accurate as possible, but accurate models are not necessarily useful or interesting and other measures such as simplicity and novelty are also important. In both data clustering and association rule discovery there is no class attribute and the data is thus unlabelled. For those two approaches patterns are learned along one of the two dimensions of the database, that is, the attribute dimension and the instance dimension. Specifically, data clustering involves identifying which data instances belong together in natural groups or clusters, whereas association rule discovery learns relationships among the attributes.

The operations research community has made significant contributions to the field of data mining and in particular to the design and analysis of data mining algorithms. Early contributions include the use of mathematical programming for both classification (Mangasarian, 1965), and clustering (Vinod, 1969; Rao, 1971), and the growing popularity of data mining has motivated a relatively recent increase of interest in this area (Bradley et al., 1999; Padmanabhan and Tuzhilin, 2003). Mathematical programming formulations now exist for a range of data mining problems, including attribute selection, classification, and data clustering. Metaheuristics have also been introduced to solve data mining problems. For example, attribute selection has been done using simulated annealing (Debuse and Rayward-Smith, 1997), genetic algorithms (Yang and Honavar, 1998) and the nested partitions method (Olafsson and Yang, 2004). However, the intersection of OR and data mining is not limited to algorithm design and data mining can play an important role in many OR applications. Vast amount of data is generated in both traditional application areas such as production scheduling (Li and Olafsson, 2005), as well as newer areas such as customer relationship management (Padmanabhan and Tuzhilin, 2003) and personalization (Murthi and Sarkar, 2003), and both data mining and traditional OR tools can be used to better address such problems.

In this paper, we present a survey of operations research and data mining, focusing on both of the abovementioned intersections. The discussion of the use of operations research techniques in data mining focuses on how numerous data mining problems can be formulated and solved as optimization problems. We do this using a range of optimization methodology, including both metaheuristics and mathematical programming. The application part of this survey focuses on a particular type of applications, namely two areas related to electronic services: customer relationship management and personalization. The intention of the paper is not to be a comprehensive survey, since the breadth of the topics would dictate a far lengthier paper. Furthermore, many excellent surveys already exist on specific data mining topics such as attribute selection, clustering, and support vector machine. The primary goals of this paper, on the other hand, are to illustrate the range of intersections of the two fields of OR and data mining, give some detailed examples of research that we believe illustrates the synergy well, provide references to other important work in the area, and finally suggest some directions for future research in the field.

2. Optimization methods for data mining

A key intersection of data mining and operations research is in the use of optimization algorithms, either directly applied as data mining algorithms, or used to tune parameters of other algorithms. The literature in this area goes back to the seminal work of Mangasarian (1965) where the problem of separating two classes of points was formulated as a linear program. This has continued to be an active research area ever since this time and the interest has grown rapidly over the past few years with the increased popularity of data mining (see e.g., Glover, 1990; Mangasarian, 1994; Bennett and Bredensteiner, 1999; Boros et al., 2000; Felici and Truemper, 2002; Street, 2005). In this section, we will briefly review different types of optimization methods that have been commonly used for data mining, including the use of mathematical programming for formulating support vector machines, and metaheuristics such as genetic algorithms.

2.1. Mathematical programming and support vector machines

One of the well-known intersections of optimization and data mining is the formulation of support vector machines (SVM) as optimization problems. Support vector machines trace their origins to the seminal work of Vapnik and Lerner (1963) but have only recently received the attention of much of the data mining and machine learning communities.

In what appears to be the earliest mathematical programming work related to this area, Mangasarian (1965) shows how to use linear programming to obtain both linear and non-linear discrimination models between separable data points or instances and several authors have since built on this work. The idea of obtaining a linear discrimination is illustrated in Fig. 1. The problem here is to determine a best model for separating the two classes. If the data can be separated by a hyperplane H as in Fig. 1, the problem can be solved fairly easily. To formulate it mathematically, we assume that the class attribute y_i takes two values -1 or $+1$. We assume that all attributes other than the class attribute are real valued and denote the training data, consisting of n instances, as $\{(\mathbf{a}_j, y_j)\}$, where $j = 1, 2, \ldots, n$, $y_j \in \{-1, +1\}$ and $\mathbf{a}_j \in \mathbf{R}^m$. If a separating hyperplane exists then there are in general many such planes and we define the optimal separating hyperplane as the one that maximizes the sum of the distances from the plane to the closest positive example and the closest negative example. To learn this optimal plane we first form the convex hulls of each of the two data sets (the positive and the negative examples), find the closest points $\mathbf{c}$ and $\mathbf{d}$ in the convex hulls, and then let the optimal hyperplane be the plane that bisects the straight line between $\mathbf{c}$ and $\mathbf{d}$. This can be formulated as a quadratic assignment problem (QAP):

$$\begin{aligned} &\min \frac{1}{2} t\|\mathbf{c} - \mathbf{d}\|^2 \\ &\text{s.t } \mathbf{c} = \sum_{i:y_i=+1} \alpha_i \mathbf{a}_i \\ &\quad \mathbf{d} = \sum_{i:y_i=-1} \alpha_i \mathbf{a}_i \\ &\quad \sum_{i:y_i=+1} \alpha_i \mathbf{a}_i = 1 \\ &\quad \sum_{i:y_i=-1} \alpha_i \mathbf{a}_i = 1 \\ &\quad \alpha_i \geqslant 0. \end{aligned} \tag{1}$$

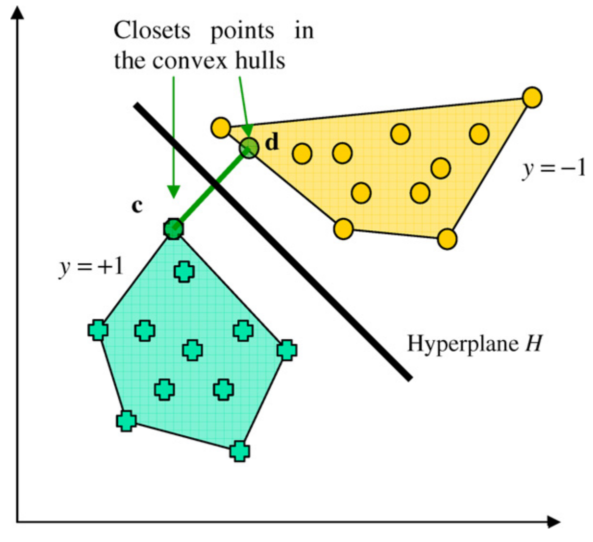

Fig. 1. A separating hyperplane to discriminate two classes.

Note that the hyperplane H can also be defined in terms of its unit normal $\mathbf{w}$ and its distance b from the origin (see Fig. 1). In other words, $H = \{\mathbf{x} \in \mathbf{R}^m : \mathbf{x} \cdot \mathbf{w} + b = 0\}$, where $\mathbf{x} \cdot \mathbf{w}$ is the dot product between those two vectors. For an intuitive idea of support vectors and support vector machines we can imagine that two hyperplanes, parallel to the original plane H and thus having the same normal, are pushed in either direction until the convex hull of the sets of all instances with each classification is encountered. This will occur at certain instances, or vectors, that are hence called the support vectors (see Fig. 2). This intuitive procedure is captured mathematically by requiring the following constraints to be satisfied:

$$\begin{aligned} &\mathbf{a}_i \cdot \mathbf{w} + b \geqslant +1, \quad \forall i : y_i = +1, \\ &\mathbf{a}_i \cdot \mathbf{w} + b \leqslant -1, \quad \forall i : y_i = -1. \end{aligned} \tag{2}$$

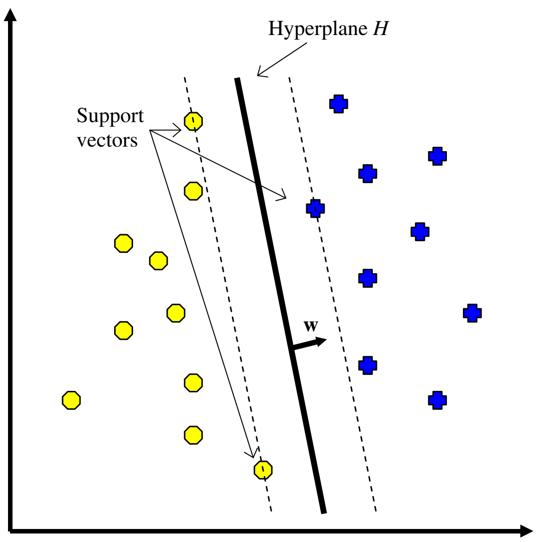

Fig. 2. Illustration of a support vector machine (SVM).

With this formulation the distance between the two planes, called the margin, is readily seen to be $2/\|\mathbf{w}\|$ and the optimal plane can thus be found by solving the following mathematical optimization problem that maximizes the margin:

$$\begin{aligned} &\max_{\mathbf{w},b}\|\mathbf{w}\|^2\\ &\text{subject to } \mathbf{a}_i\cdot\mathbf{w}+b \geqslant +1, \quad \forall i: y_i=+1 \qquad (3)\\ &\qquad\qquad \mathbf{a}_i\cdot\mathbf{w}+b \leqslant -1, \quad \forall i: y_i=-1. \end{aligned}$$

When the data is non-separable this problem will have no feasible solution and the constraints for the existence of the two hyperplanes must be relaxed. One way of accomplishing this is by introducing error variables ε_j for each instance $\mathbf{a}_j$, $j=1,2,\ldots,n$. Essentially these variables measure the violation of each instance and using these variables the following modified constraints are obtained for problem (3):

$$\begin{aligned} &\mathbf{a}_i\cdot\mathbf{w}+b \geqslant +1-\varepsilon_i, \quad \forall i: y_i=+1\\ &\mathbf{a}_i\cdot\mathbf{w}+b \leqslant -1+\varepsilon_i, \quad \forall i: y_i=-1 \qquad (4)\\ &\varepsilon_i \geqslant 0, \quad \forall i. \end{aligned}$$

As the variables ε_j represent the training error, the objective could be taken to minimize $\|\mathbf{w}\|/2 + C\cdot\sum_j \varepsilon_j$, where C is constant that measures how much penalty is given. However, rather than formulating this problem directly it turns out to be convenient to formulate the following dual program:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{a}_i\cdot\mathbf{a}_j\\ \text{subject to } & 0\leqslant \alpha_i \leqslant C \qquad (5)\\ & \sum_i \alpha_i y_i = 0. \end{aligned}$$

The solution to this problem are the dual variables $\boldsymbol{\alpha}$ and to obtain the primal solution, that is, the model classifying instances defined by the normal of the hyperplane, we calculate

$$\mathbf{w} = \sum_{i:\mathbf{a}_i \text{ support vector}} \alpha_i y_i \mathbf{a}_i. \qquad (6)$$

The benefit of using the dual is that the constraints are much simpler and easier to handle and that the training data only enters in (5) through the dot product $\mathbf{a}_i\cdot\mathbf{a}_j$. This latter point is important for extending the approach to non-linear model.

Requiring a hyperplane or a linear discrimination of points is clearly too restrictive for most problems. Fortunately, the SVM approach can be extended to non-linear models in a very straightforward manner using what is called kernel functions $K(\mathbf{x},\mathbf{y})=\phi(\mathbf{x})\cdot\phi(\mathbf{y})$, where $\phi:\mathbf{R}^m\to H$ is a mapping from the m-dimensional Euclidean space to some Hilbert space H. This approach was introduced to the SVM literature by Cortes and Vapnik (1995) and it works because the data $\mathbf{a}_j$ only enters the dual via the dot product $\mathbf{a}_i\cdot\mathbf{a}_j$, which can thus be replaced with $K(\mathbf{a}_i,\mathbf{a}_j)$. The choice of kernel determines the model. For example, to fit a p degree polynomial the kernel can be chosen as $K(\mathbf{x},\mathbf{y})=(\mathbf{x}\cdot\mathbf{y}+1)^p$. Many other choices have been considered in the literature but we will not explore this further here. Detailed expositions of SVM can be found in the book by Vapnik (1995) and in the survey papers of Burges (1998) and Bennett and Campbell (2000).

2.2. *Metaheuristics for combinatorial optimization*

Many optimization problems that arise in data mining are discrete rather than continuous and numerous combinatorial optimization formulations have been suggested for such problems. This includes for example attribute selection, that is, the problem of determining the best set of attributes to be used by the learning algorithm (see Section 3.1), determining the optimal structure of a Bayesian network in classification (see Section 3.1.2), and finding the optimal clustering of data instances (see Section 3.2). In particular, many metaheuristic approaches have been proposed to address such data mining problems.

Metaheuristics are the preferred method over other optimization methods primarily when there is a need to find good heuristic solutions to complex optimization problems with many local optima and little inherent structure to guide the search (Glover and Kochenberger, 2003). Many such problems arise in the data mining context. The metaheuristic approach to solving such problem is to start by obtaining an initial solution or an initial set of solutions and then initiating an improving search guided by certain principles. The structure of the search has many common elements across various methods. In each step of the search algorithm, there is always a solution $\mathbf{x}_k$ (or a set of solutions) that represents the current state of the algorithm. Many metaheuristics are solution-to-solution search methods, that is, $\mathbf{x}_k$ is a single solution or point $\mathbf{x}_k \in \mathbf{X}$ in some solution space $\mathbf{X}$, corresponding to the feasible region. Others are set-based, that is, in each step $\mathbf{x}_k$ represents a set of solutions $\mathbf{x}_k \subseteq \mathbf{X}$. However, the basic structure of the search remains the same regardless of whether the metaheuristics is solution-to-solution or set-based.

The reason for the meta-prefix is that metaheuristics do not specify all the details of the search, which can thus be adapted by a local heuristic to a specific data mining application. Instead, they specify general strategies to guide specific aspects of the search. For example, tabu search uses a list of solutions or moves called the tabu list, which ensures the search does not revisit recent solutions or becomes trapped in local optima. The tabu list can thus be thought of as a restriction of the neighborhood. On the other hand, methods such as genetic algorithm specify the neighborhood as all solutions that can be obtained by combining the current solutions through certain operators. Other methods, such as simulated annealing, do not specify the neighborhood in any way but rather specify an approach to accepting or rejecting solutions that allows the method to escape local optima. Finally, the nested partitions method is an example of a set-based method that selects candidate solutions from the neighborhood with probability distribution that adapts as the search progresses to make better solutions be selected with higher probability.

All metaheuristics can be thought of to share the elements of selecting candidate solution(s) from a neighborhood of the current solution(s) and then either accepting or rejecting the candidate(s). With this perspective, each metaheuristics is thus defined by specifying one or more of these elements but allowing others to be adapted to the particular application. This may be viewed as both strength and a liability. It implies that we can take advantage of special structure for each application but it also means that the user must specify those aspects, which can be complicated. For the remainder of this section we briefly discuss a few of the most common metaheuristics and discuss how they fit within this framework.

One of the earliest metaheuristics is simulated annealing (Kirkpatrick et al., 1983), which is motivated by the physical annealing process, but within the framework here simply specifies a method for determining if a solution should be accepted. As a solution-to-solution search method, in each step it selects a candidate $\mathbf{x}^c$ from the neighborhood $N(\mathbf{x}_k)$ of the current solution $\mathbf{x}_k \in \mathbf{X}$. The definition of the neighborhood is determined by the user. If the candidate is better than the current solution it is accepted. If it is worse it is not automatically rejected but rather accepted with probability $P[\text{Accept } \mathbf{x}^c] = e^{f(x_k)-f(x^c)/T_k}$, where $f: \mathbf{X} \to \mathbf{R}$ is a real valued objective function to be minimized and T_k is a parameter called the temperature. Clearly, the probability of acceptance is high if the performance difference is small and T_k is large. The key to simulated annealing is to specify a cooling schedule $\{T_k\}_{k=1}^{\infty}$ by which the temperature is reduced so that initially inferior solutions are selected with a high enough probability so local optimal are escaped but eventually it becomes small enough so that the algorithm converges. Simulated annealing has for example been used to solve the attribute selection problem in data mining (Debuse and Rayward-Smith, 1997, 1999).

Other popular solution-to-solution metaheuristics include tabu search, the greedy randomized adaptive search procedure (GRASP) and the variable neighborhood search (VNS). The defining characteristic of tabu search is in how solutions are selected from the neighborhood. In each step of the algorithm, there is a list L_k of solutions that were recently visited and are therefore tabu. The algorithm looks through all of the solutions of the neighborhood that are not tabu and selects the best one. The defining property of GRASP is its multistart approach that initializes several local search procedures from different starting points. The advantage of this is that the search becomes more global, but on the other hand each search cannot use what the other searches have learned, which introduces some inefficiency. The VNS is interesting in that it uses an adaptive neighborhood structure that changes based on the performance of the solutions that are evaluated. More information on tabu search can be found in Glover and Laguna (1997), GRASP is discussed in Resende and Ribeiro (2003), and for an introduction to the VNS approach we refer the reader to Hansen and Mladenovic (1997).

Several metaheuristics are set-based or population based rather than solution-to-solution. This includes genetic algorithms and other evolutionary approaches, as well as scatter search and the nested partitions method. The most popular metaheuristic used in data mining is in fact genetic algorithm and its variants. As an approach to global optimization genetic algorithms (GA) have been found to be applicable to optimization problems that are intractable for exact solutions by conventional methods (Holland, 1975; Goldberg, 1989). It is a set-based search algorithm where at each iteration it simultaneously generates a number of solutions. In each step, a subset of the current set of solutions is selected based on their performance and these solutions are combined into new solutions. The operators used to create the new solutions are survival, where a solution is carried to the next iteration without change, crossover, where the properties of two solutions are combined into one, and mutation, where a solution is modified slightly. The same process is then repeated with the new set of solutions. The crossover and mutation operators depend on the representation of the solution but not on the evaluation of its performance. The selection of solutions, however, does depend on the performance. The general principle is that high performing solutions (which in genetic algorithms are referred to as fit individuals) should have a better chance of both surviving and being allowed to create new solutions through crossover. For genetic algorithms and other evolutionary methods the defining element is the innovative manner in which the crossover and mutation operators define a neighborhood of the current solution. This allows the search to quickly and intelligently traverse large parts of the solution space. In data mining genetic and evolutionary algorithms have been used to solve a host of problems, including attribute selection (Yang and Honavar, 1998; Kim et al., 2000) and classification (Fu et al., 2003a,b; Larrañaga et al., 1996; Sharpe and Glover, 1999).

Scatter search is another metaheuristic related to the concept of evolutionary search. In each step a scatter search algorithm considers a set of solutions called the reference set. Similar to the genetic algorithm approach these solutions are then combined into a new set. However, as opposed to the genetic operators, in scatter search the solutions are combined using linear combinations, which thus define the neighborhood. For references on scatter search we refer the reader to Glover et al. (2003).

Introduced by Shi and Olafsson (2000), the nested partition method (NP) is another metaheuristic for combinatorial optimization. The key idea behind this method lies in systematically partitioning the feasible region into subregions, evaluating the potential of each region, and then focusing the computational effort to the most promising region. This process is carried out iteratively with each partition nested within the last. The computational effectiveness of the NP method relies heavily on the partitioning, which if carried out in a manner such that good solutions are close together can reach a near optimal solution very quickly. In data mining, the NP algorithm has been used for attribute selection (Olafsson and Yang, 2004; Yang and Olafsson, 2006), and clustering (Kim and Olafsson, 2004).

3. The data mining process

As described in the introduction, data mining involves using an inductive algorithm to learn previously unknown patterns from a large database. But before the learning algorithm can be applied a great deal of data preprocessing must usually be performed. Some authors distinguish this from the inductive learning by referring to the whole process as knowledge discovery and reserve the term data mining for only the inductive learning part of the process. As stated earlier, however, we refer to the whole process as data mining. Typical steps in the process include the following:

- As for other data analyses projects data mining starts by defining the business or scientific objectives and formulating this as a data mining problem.
- Given the problem to be addressed, the appropriate data sources need to be identified and the data integrated and preprocessed to make it appropriate for data mining (see Section 3.1)

- Once the data has been prepared the next step is to generate previously unknown patterns from the data using inductive learning. The most common types of patterns are classification models (see Section 3.1.2), natural cluster of instances (see Section 3.2), and association rules describing relationships between attributions (see Section 3.3).
- The final steps are to validate and then implement the patterns obtained from the inductive learning.

In the following sections, we describe several of the most important parts of this process and focus specifically on how optimization methods can be used for the various parts of the process.

3.1. Data preprocessing and exploratory data mining

In any data mining application the database to be mined may contain noisy or inconsistent data, some data may be missing and in almost all cases the database is large. Data preprocessing addresses each of those issues and includes such preliminary tasks as data cleaning, data integration, data transformation, and data reduction. Also applied in the early stages of the process exploratory data mining involves discovering patterns in the data using summary statistics and visualization. Optimization and other OR tools are relevant to both data preprocessing tasks and exploratory data mining and in this section we illustrate this through one particular task from each category, namely attribute selection and data visualization.

3.1.1. Attribute selection

Attribute selection is an important problem in data mining. This involves a process for determining which attributes are relevant in that they predict or explain the data, and conversely which attributes are redundant or provide little information (Liu and Motoda, 1998). Doing attribute selection before a learning algorithm is applied has numerous benefits. By eliminating many of the attributes it becomes easier to train other learning methods, that is, computational time of the induction is reduced. Also, the resulting model may be simpler, which often makes it easier to interpret and thus more useful in practice. It is also often the case that simple models are found to generalize better when applied for prediction. Thus, a model employing fewer attributes is likely to score higher on many interestingness measures and may even score higher in accuracy. Finally, discovering which attributes should be kept, that is identifying attributes that are relevant to the decision making, often provides valuable structural information and is therefore important in its own right.

The literature on attribute selection is extensive and many attribute selection methods are based on applying an optimization approach. As a recent example, Olafsson and Yang (2004) formulate the attribute selection problem as a simple combinatorial optimization problem with the following decision variables:

$$x_i = \begin{cases} 1 & \text{if the } i\text{th feature is included} \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

for $i = 1, 2, \ldots, m$. The optimization problem is then

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & K_{\min} \leqslant \sum_{i=1}^{m} x_i \leqslant K_{\max}, \\ & x_i \in \{0, 1\} \end{aligned} \tag{8}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ and $K_{\min}$ and $K_{\min}$ are some minimum and maximum number of attributes to be selected. A key issue is the selection of the objective function and there is no single method for evaluating the quality of attributes that works best for all data mining problems. Some methods evaluate the quality of each attribute individually, that is,

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(x_i), \tag{9}$$

whereas others evaluate the quality of the entire subset together, that is, $f(\mathbf{x}) = f(X)$, where $X = \{i : x_i = 1\}$ is the subset of selected attributes. In Olafsson and Yang (2004) the authors use the nested partitions method of Shi and Olafsson (2000) to solve this problem using multiple objective functions of both types described above and show that such an optimization approach is very effective. In Yang and Olafsson (2006) the authors improve these results by developing an adaptive version of the algorithm, which in each step uses a small random subset of all the instances. This is important because data mining usually deals with very large number of instances and scalability with respect to number of instances is therefore a critical issue. Other optimization-based methods that have been applied to this problem include genetic algorithms

(Yang and Honavar, 1998), evolutionary search (Kim et al., 2000), simulated annealing (Debuse and Rayward-Smith, 1997), branch-and-bound (Narendra and Fukunaga, 1977), logical analysis of data (Boros et al., 2000), and mathematical programming (Bradley et al., 1998).

3.1.2. Data visualization

As for most other data analysis, the visualization of data plays an important role in data mining. This is a difficult problem since the data is usually high dimensional, that is, the number m of attributes is large, whereas the data can only be visualized in two or three dimensions. While it is possible to visualize two or three attributes at a time a better alternative is often to map the data to two or three dimensions in a way that preserves the structure of the relationships (that is, distances) between instances. This problem has traditionally been formulated as a non-linear mathematical programming problem (Borg and Groenen, 1997).

As an alternative to the traditional formulation, Abbiw-Jackson et al. (2006) recently provide the following quadratic assignment problem (QAP) formulation. Given n instances in $\mathbf{R}^m$ and a matrix $\mathbf{D}^{\text{old}} \in \mathbf{R}^n \times \mathbf{R}^n$ measuring the distance between those instances, find the optimal assignment of those instances to a lattice N in $\mathbf{R}^q$, $q = 2, 3$. The decision variables are given by

$$x_{ik} = \begin{cases} 1, & \text{if the } i\text{th instance is assigned} \\ & \text{to lattice point } k \in N, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

With this assignment, there is a new distance matrix $\mathbf{D}^{\text{new}} \in \mathbf{R}^q \times \mathbf{R}^q$ in the $q = 2$ or 3 dimensional space, and the mathematical program can be written as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k\in N} \sum_{l\in N} F(\mathbf{D}^{\text{old}}_{i,j}, \mathbf{D}^{\text{new}}_{i,j}) x_{ik} x_{jl} \\ \text{subject to} \quad & \sum_{k\in N} x_{ik} = 1, \quad \forall i \\ & x_{ik} \in \{0, 1\}, \end{aligned} \tag{11}$$

where F is a function of the deviation between the differences between the instances in the original space and the new q-dimensional space. Any solution method for the quadratic assignment method can be used, but Abbiw-Jackson et al. (2006) propose a local search heuristic that take the specific objective function into account and compare the results to the traditional non-linear mathematical programming formulations. They conclude that the QAP formulation provides similar results and importantly tends to perform better for large problems.

3.2. Classification

Once the data has been preprocessed a learning algorithm is applied, and one of the most common learning tasks in data mining is classification. Here there is a specific attribute called the class attribute that can take a given number of values and the goal is to induce a model that can be used to discriminate new data into classes according to those values. The induction is based on a labeled training set where each instance is labeled according to the value of the class attribute. The objective of the classification is to first analyze the training data and develop an accurate description or a model for each class using the attributes available in the data. Such class descriptions are then used to classify future independent test data or to develop a better description for each class. Many methods have been studied for classification, including decision tree induction, support vector machines, neural networks, and Bayesian networks (Fayyad et al., 1996; Weiss and Kulikowski, 1991).

Optimization is relevant to many classification methods and support vector machines have already been mentioned in Section 2.2 above. In this section, we focus on three additional popular classification approaches, namely decision tree induction, Bayesian networks, and neural networks.

3.2.1. Decision trees

One of the most popular techniques for classification is the top-down induction of decision trees. One of the main reason behind their popularity appears to be their transparency, and hence relative advantage in terms of interpretability. Another advantage is the ready availability of powerful implementations such as CART (Breiman et al., 1984) and C4.5 (Quinlan, 1993). Most decision tree induction algorithms construct a tree in a top-down manner by selecting attributes one at a time and splitting the data according to the values of those attributes. The most important attribute is selected as the top split node, and so forth. For example, in C4.5 attributes are chosen to maximize the information gain ratio in the split (Quinlan, 1993). This is an entropy measure designed to increase the average class purity of the resulting subsets. Algorithms such as C4.5 and CART are computationally efficient and have proven very successful in practice. However, the fact that they are limited to constructing axis-parallel separating planes limits their effectiveness in applications where some combination of attributes is highly predictive of the class (Lee and Olafsson, 2006).

Mathematical optimization techniques have been applied directly in the optimal construction of decision boundaries in decision tree induction. In particular, Bennett (1992) introduced an extension of linear programming techniques to decision tree construction, although this formulation is limited to two-class problems. In recent work, Street (2005) presents a new algorithm for multi-category decision tree induction based on non-linear programming. The algorithm, termed Oblique Category SEParation (OC-SEP), shows improved generalization performance on several real-world data sets.

One of the limitations of most decision tree algorithms is that they are known to be unstable. This is especially true dealing with a large data set where it can be impractical to access all data at once and construct a single decision tree (Fu et al., 2003a). To increase interpretability, it is necessary to reduce the tree sizes and this can make the process even less stable. Finding the optimal decision tree can be treated as a combinatorial optimization problem but this is known to be an NP-complete problem and heuristics such as those discussed in Section 2.2 above must be applied. Kennedy et al. (1997) first developed a genetic algorithm for optimizing decision trees. In their approach, a binary tree is represented by a number of unit subtrees, each having a root node and two branches. In more recent work, Fu et al. (2003a,b, 2006) also use genetic algorithms for this task. Their method uses C4.5 to generate *K* trees as the initial population, and then exchanges the subtrees between trees (crossover) or within the same tree (mutation). At the end of a generation, logic checks and pruning are carried out to improve the decision tree. They show that the resulting tree performs better than C4.5 and the computation time only increases linearly as the size of the training and scoring combination increases. Furthermore, creating each tree only requires a small percent of data to generate high-quality decision trees. All of the above approaches use some function of the tree accuracy for the genetic algorithm fitness function. In particular, Fu et al. (2003a) use the average classification accuracy directly, whereas Fu et al. (2003b) use a distribution for the accuracy that enables them to account for the user's risk tolerance. This is further extended in Fu et al. (2006) where the classification is modeled using a loss function that then becomes the fitness function of the genetic algorithm. Finally, in other related work Dhar et al. (2000) use an adaptive resampling method where instead of using a complete decision tree as the chromosomal unit, a chromosome is simply a rule, that is, any complete path from the root node of the tree to a leaf node.

When using genetic algorithm to optimize the three, there is ordinarily no method for adequately controlling the growth of the tree, because the genetic algorithm does not evaluate the size of the tree. Therefore, during the search process the tree may become overly deep and complex or may settle to a too simple tree. To address this, Niimi and Tazaki (2000) combine genetic programming with association rule algorithm for decision tree construction. In this approach rules generated by the Apriori association rule discovery algorithm (Agrawal et al., 1993) are taken as the initial individual decision trees for a subsequent genetic programming algorithm.

Another approach to improve the optimization of the decision tree is to improve the fitness function used by the genetic algorithm. Traditional fitness functions use the mean accuracy as the performance measure. Fu et al. (2003b) investigate the use of various percentiles of the distribution of classification accuracy in place of the mean and developed a genetic algorithm that simultaneously considers two fitness criteria. Tanigawa and Zhao (2000) include the tree size in the fitness function in order to control the tree's growth. Also, the utilization of a fitness function based on the J-Measure, which determines the information content of a tree, can give a preference criterion to find the decision tree that classifies a set of instances in the best way (Folino et al., 2001).

3.2.2. Bayesian networks

The popular naïve Bayes method is another simple but yet effective classifier. This method learns the conditional probability of each attribute given the class label from the training data. Classification is then done by applying Bayes rule to compute the probability of a class value given the particular instance and predicting the class value with the highest probability. In general this would require estimating the marginal probabilities of every attribute combination, which is not feasible, especially when the number of attributes is large and there may be few or no observations (instances) for some of the attribute combinations. Thus, a strong independence assumption is made, that is, all the attributes are assumed conditionally independent given the value of the class attribute. Given this assumption, only the marginal probabilities of each attribute given the class need to be calculated. However, this assumption is clearly unrealistic and Bayesian networks relax it by explicitly modeling dependencies between attributes.

A Bayesian network is a directed acyclic graph G that models probabilistic relationships among a set of random variables $\mathbf{U} = \{X_1, \ldots, X_m\}$, where each variable in $\mathbf{U}$ has specific states or values (Jensen, 1996). As before, m denotes the number of attributes. Each node in the graph represents a random variable, while the edges capture the direct dependencies between the variables. The network encodes the conditional independence relationships that each node is independent of its non-descendants given its parents (Castillo et al., 1997; Pernkopf, 2005). There are two key optimization-related issues when using Bayesian networks. First, when some of the nodes in the network are not observable, that is, there is no data for the values of the attributes corresponding to those nodes, finding the most likely values of the conditional probabilities can be formulated as a non-linear mathematical program. In practice this is usually solved using a simple steepest descent approach. The second optimization problem occurs when the structure of the network is unknown, which can be formulated as a combinatorial optimization problem.

The problem of learning the structure of a Bayesian network can be informally stated as follows. Given a training set $\mathbf{A} = \{u_1, u_2, \ldots, u_n\}$ of n instances of $\mathbf{U}$ find a network that best matches $\mathbf{A}$. The common approach to this problem is to introduce an objective function that evaluates each network with respect to the training data and then to search for the best network according to this function (Friedman et al., 1997). The key optimization challenges are choosing the objective function and determining how to search for the best network. The two main objective functions commonly used to learn Bayesian networks are the Bayesian scoring function (Cooper and Herskovits, 1992; Heckerman et al., 1995), and a function based on the minimal description length (MDL) principle (Lam and Bacchus, 1994; Suzuki, 1993). Any metaheuristic can be applied to solve the problem. For example, Larrañaga et al. (1996) have done work on using genetic algorithms for learning Bayesian networks.

3.2.3. Neural networks

Another popular approach for classification is neural networks. Neural networks have been extensively studied in the literature and an excellent review of the use of feed-forward neural networks for classification is given by Zhang (2000). The inductive learning of neural networks from data is referred to as training this network, and the most popular method of training is back-propagation (Rumelhart and McClelland, 1986). It is well known that back-propagation can be viewed as an optimization process and since this has been studied in detailed elsewhere we only briefly review the primary connection with optimization here.

A neural network consists of at least three layers of nodes. The input layer consists of one node for each of the independent attributes. The output layer consists of node(s) for the class attribute(s), and connecting these layers is one or more intermediate layers of nodes that transform the input into an output. When connected, these layers of nodes make up the network we refer to as a neural net. The training of the neural network involves determining the parameters for this network. Specifically, each arc connecting the nodes in this network has certain associated weight and the values of those weights determine how the input is transformed into an output. Most neural network training methods, including back-propagation, are inherently an optimization processes. As before, the training data consists of values for some input attributes (input layer) along with the class attribute (output layer), which is usually referred to as the target value of the network. The optimization process seeks to determine the arc weights in order to reduce some measure of error (normally, minimizing squared error) between the actual and target outputs (Ripley, 1996).

Since the weights in the network are continuous variables and the relationship between the input and the output is highly non-linear, this is a non-linear continuous optimization problem or a non-linear programming problem (NLP). Any appropriate NLP algorithm could therefore be applied to train a neural network, but in practice a simple steepest-descent approach is most often applied (Ripley, 1996). This does not assure that the global optimal solution has been found, but rather terminates at the first local optimum that is encountered. However, due to the size of the problems involved the speed of the optimization algorithm is usually imperative.

This section illustrates how optimization plays significant role in classification. As shown in Section 2.1, the classification problem itself can be formulated as a mathematical programming problem, and as demonstrated in this section it also plays an important role in conjunction with other classification methods. This can both be to optimize the output of other classification algorithms, such as the optimization of decision trees, and to optimize parameters utilized by other classification algorithms, such as finding the optimal structure of a Bayesian network. Although considerable work has been done already in this area there are still many unsolved problems for the operations research community to address.

3.3. Clustering

When the data is unlabelled and each instance does not have a given class label the learning task is called unsupervised. If we still want to identify which instances belong together, that is, form natural clusters of instances, a clustering algorithm can be applied (Jain et al., 1999; Kaufman and Rousseeuw, 1990). Such algorithms can be divided into two categories: hierarchical clustering and partitional clustering. In hierarchical clustering all of the instances are organized into a hierarchy that describes the degree of similarity between those instances. Such representation may provide a great deal of information and many algorithms have been proposed. Partitional clustering, on the other hand, simply creates one partition of the data where each instance falls into one cluster. Thus, less information is obtained but the ability to deal with large number of instances is improved.

As appears to have been first pointed out by Vinod (1969), the partitional clustering problem can be formulated as an optimization problem. The key issues are how to define the decision variables and how to define the objective functions, neither of which has a universally applicable answer. In clustering, the most common objectives are to minimize the difference of the instances in each cluster (compactness), maximize the difference between instances in different clusters (separation), or some combination of the two measures. However, other measures may also be of interest and adequately assessing cluster quality is a major unresolved issue (Estevill-Castro, 2002; Grabmeier and Rudolph, 2002; Osei-Bryson, 2005). A detailed discussion of this issue is outside the scope of the paper and most of the work that applies optimization to clustering focuses on some variant of the compactness measure.

In addition to the issue of selecting an appropriate measure of cluster quality as the objective function there is no generally agreed upon manner in which a clustering should be defined. One popular way to define a data clustering is to let each cluster be defined by its center point $\mathbf{c}_j \in \mathbf{R}^m$ and then assign every instance to the closest center. Thus, the clustering is defined by a $m \times k$ matrix $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k)$. This is for example done by the classic and still popular k-means algorithm (MacQueen, 1967). K-means is a simple iterative algorithm that proceeds as follows. Starting with randomly selected instances as centers each instance is first assigned to the closest center. Given those assignments, the cluster centers are recalculated and each instance again assigned to the closest center. This is repeated until no instance changes clusters after the centers are recalculated, that is, the algorithm converges to a local optimum.

Much of the work on optimization formulations uses the idea of defining a clustering by fixed number of centers. This is true of the early work of Vinod (1969) where the author provides two integer programming formulations of the clustering problem. For example, in the first formulation the decision variable is defined as an indicator of the cluster to which each instance is assigned:

$$x_{ij} = \begin{cases} 1 & \text{if the } i\text{th instance is assigned} \\ & \text{to the } j\text{th cluster,} \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

and the objective is to minimize the total cost of the assignment, where w_{ij} is some cost of assigning the ith instance to the jth cluster:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{k} x_{ij} = 1, \quad i = 1,2,\ldots,n \\ & \sum_{i=1}^{n} x_{ij} \geqslant 1, \quad j = 1,2,\ldots,k. \end{aligned} \tag{13}$$

Note that the constraints assure that each instance is assigned to exactly one cluster and that each cluster has at least one instance (all the clusters are used). Also note that if the assignments (12) are known then the cluster centers can be calculated by averaging the assigned instances. Using the same definition of the decision variables, Rao (1971) provides additional insights into the clustering problem and suggests improved integer programming formulations, with the objective taken as both minimizing the within cluster sum of squares and minimizing the maximum distance within clusters. Both of those objective functions can be viewed as measures of cluster compactness.

More recently, Bradley et al. (1996) and Bradley and Mangasarian (2000) also formulated the problem of identifying cluster centers as a mathematical programming problem. As before, a set $\mathbf{A}$ of n training instances $\mathbf{A}_i \in \mathbf{R}^n$ is given, and we assume a fixed number of k clusters. Given this scenario, Bradley et al. (1996) use dummy vectors $\mathbf{d}_{ij} \in \mathbf{R}^n$ to formulate the following linear program to find the k cluster centers $\mathbf{c}_j$ that minimize the 1-norm from each instance to the nearest center:

$$\begin{aligned} \min_{\mathbf{c},\mathbf{d}} \quad & \sum_{i=1}^{n} \min\{\mathbf{e}^{\mathrm{T}}\mathbf{d}_{ij}\} \\ \text{s.t.} \quad & -\mathbf{d}_{ij} \leqslant \mathbf{A}_i^T - \mathbf{c}_j \leqslant \mathbf{d}_{ij}, \quad i=1,2,\ldots,n;\ j=1,2,\ldots,k. \end{aligned} \tag{14}$$

By using the 1-norm instead of more usual 2-norm this is a linear program, which can be solved efficiently even for very large problem. As for the other formulations discussed above a limitation to this program is that it focuses exclusively on optimizing the cluster compactness, which in this case means to find the cluster centers such that each instance in the cluster is as close as possible to the center. The solution may therefore be a cluster set where the clusters are not well separated.

In Bradley and Mangasarian (2000), the authors take a different definition of a clustering and instead of finding the best centers identify the best cluster planes:

$$P_j = \{x \in \mathbf{R}^m | \mathbf{x}^{\mathrm{T}} \cdot \mathbf{w}_j = \gamma_j\}, \quad j = 1,2,\ldots,k. \tag{15}$$

They propose an iterative algorithm similar to the k-means algorithm that iteratively assigns instances to the closes cluster and then given the new assignment finds the plane that minimizes the sum of squares of distances of each instance to the cluster. In other words, given the set of instances $\mathbf{A}^{(j)} \in \mathbf{R}^n$ assigned to cluster j, find $\boldsymbol{w}$ and γ that solve:

$$\begin{aligned} \min_{\mathbf{w},\gamma} \quad & \|\mathbf{A}\mathbf{w} - \mathbf{e}\gamma\|_2^2 \\ \text{s.t.} \quad & \mathbf{w}^{\mathrm{T}} \cdot \mathbf{w} = 1. \end{aligned} \tag{16}$$

Indeed it is not necessary to solve (16) using traditional methods. The authors show that a solution $(\mathbf{w}_j^*, \gamma_j^*)$ can be found by letting $\mathbf{w}_j^*$ be the eigenvector corresponding to the smallest eigenvector of $(\mathbf{A}^{(j)})^{\mathrm{T}}(\mathbf{I} - \mathbf{e} \cdot \mathbf{e}^{\mathrm{T}}/n_j)\mathbf{A}^{(j)}$, where $n_j = |\mathbf{A}^{(j)}|$ is the number of instances assigned to the cluster, and then calculating $\gamma_j^* = \mathbf{e}^{\mathrm{T}}\mathbf{A}^{(j)}\mathbf{w}_j^*/n_j$.

We note from the formulations above that the clustering problem can be formulated as both an integer program, as in (13), and a continuous program, as in (14). This implies that a wide array of optimization techniques is applicable to the problem. For example, Kroese et al. (2004) recently used the cross-entropy method to solve both discrete and continuous versions of the problem. They show that although the cross-entropy method is more time consuming than traditional heuristics such as k-means the quality of the results is significantly better.

As noted by both the early work in this area (Vinod, 1969; Rao, 1971) and by more recent authors (e.g., Shmoys, 1999), when a clustering is defined by the cluster centers the clustering problem is closely related to well-known optimization problems related to set covering. In particular, the problem of locating the best clusters mirrors problems in facility location (Shmoys et al., 1997), and specifically the k-center and k-median problem. In many cases, results obtained for these problems could be directly applied to clustering in data mining. For example, Hochbaum and Shmoys (1985) proposed the following approximation algorithm for the k-center problem. Starting with any point, find the point furthest from it, then the point furthest from the first two, and so forth, until k points have been selected. The authors use duality theory to show that the performance of this heuristic is no worse than twice the performance of the optimal solution. Interpreting the points as centers, Dasgupta (2002) notes that this approach can be used directly for partitional clustering. The author also develops an extension to hierarchical clustering and derives a similar performance bound for the clustering performance of every level of the hierarchy.

It should be noted that this section has focused on one particular type of clustering, namely partitional clustering. For many of the optimization formulations it is further assumed that each cluster is defined by its center, although the (13) does not make this assumption and other formulations from, for example, Rao (1971) and Kroese et al. (2004) are more flexible. Several other approaches exist for clustering and much research on clustering for data mining has focused on the scalability of clustering algorithms (see e.g., Bradley et al., 1998; Ng and Han, 1994; Zhang et al., 1996). As another example of a partitional clustering method, the well-known EM algorithm assumes instances are drawn from one of k distributions and estimates the parameters of these distributions as well as the cluster assignments (Lauritzen, 1995). There have also been numerous hierarchical clustering algorithms proposed, which create a hierarchy, such as a dendrogram, that shows the relationship between all the instances (Kaufman and Rousseeuw, 1990). Such clustering may be either agglomerative, where instances are initially assigned to singleton clusters and are then merged based on some measure (such as the well-known single-link and complete-link algorithms), or divisive, where the instances are all assigned to one cluster that is then split iteratively. For hierarchical clustering there no single clustering that is optimal, although one may ask the question if it is possible to find a hierarchy that is optimal at every level and extend known partitional clustering results (Dasgupta, 2002).

Another clustering problem where optimization methods have been successfully applied is sequential clustering (Hwang, 1981). This problem is equivalent to the partitional clustering problem described above, except that the instances are ordered and this order must be maintained in the clustering. Hwang (1981) shows how to find optimal clusters for this problem, and in recent work Joseph and Bryson (1997) show how to find so-called w-efficient solutions to sequential clustering using linear programming. Such solutions have satisfactory rate of improvement in cluster compactness as the number of clusters increases.

Although it is well known that the clustering problem can be formulated as an optimization problem the increased popularity of data mining does not appear to have resulted in comparable resurgence of interest in use of optimization to cluster data as for classification. For example, a great deal of work has been done on closely related problems such as the set covering, k-center and k-median problems, but relatively little has still been done to investigate potential impact in the data mining task of clustering. Since the clustering problem can be formulated as both a continuous mathematical programming problem and a combinatorial optimization problem, a host of OR tools is applicable to this problem, including both mathematical programming (see Section 2.1) and metaheuristics (see Section 2.2), and most of this has not been addressed. Other fundamental issues, such as how to define a set of clusters and measure cluster quality are also still unresolved in the larger clustering community, and it is our belief that the OR community could continue to make very significant contributions by examining these issues.

3.4. Association rule mining

Clustering finds previously unknown patterns along the instance dimension. Another unsupervised learning approach is association rule discovery that aims to discover interesting correlation or other relationships among the attributes (Agrawal et al., 1993). Association rule mining was originally used for market basket analysis, where items are articles in the customer's shopping cart and the supermarket manager is looking for associations among these purchases. Basket data stores items purchased on per-transaction basis. The questions addressed by market basket analysis include how to boost the sales of a given product, what other products do discontinuing a product impact, and which products should be shelved together. Being derived from market basket analysis, association rule discovery uses the terminology of an item, which is simply an attribute – value pair, and item set, which simply refers to a set of such items.

With this terminology the process of association rule mining can be described as follows. Let $I = \{1, 2, \ldots, q\}$ be the set of all items and let $T = \{1, 2, \ldots, n\}$ be the set of transactions or instances in a database. An association rule R is an expression $A \Rightarrow B$, where the antecedent (A) and consequent (B) are both item sets, that is $A, B \subseteq I$. Each rule has an associated support and confidence. The support $\sup(R)$ of the rule is the number or percentage of instances in I containing both A and B, and this is also referred to as the coverage of the rule. The confidence of the rule R is given by

$$\mathrm{conf}(R) = \frac{\sup(A \cup B)}{\sup(A)}, \tag{17}$$

which is the conditional probability that an instance contains item set B given that it contains item set A.

Confidence is also called accuracy of the rule. Support and confidence are typically modeled as constraints for association rule mining, where users specify the minimum support $\sup_{\min}$ and minimum confidence $\mathrm{conf}_{\min}$ according to their preferences. An item set is called a frequent item set if its support is greater than this minimum support threshold.

Apriori is the best-known and original algorithm for association rule discovery (Agrawal and Srikant, 1994). The idea behind the apriori algorithm is that if an item set is not frequent in the database then any superset of this item set is not frequent in the same database. There are two phases to the inductive learning: (a) first find all frequent item sets, and (b) then generate high confidence rules from those sets. The apriori algorithm generates all frequent item sets by making multiple passes over the data. In the first pass it determines whether 1-item sets are frequent or not according to their support. In each subsequent pass it starts with those item sets found to be frequent in the previous pass. It uses these frequent items sets as seed sets to generate super item sets, called candidate item sets, by only adding one more item. If the super item set meets the minimum support then it is actually frequent. After frequent item sets' generation, for each final frequent item set it checks all single-consequent rules. Only those single consequent rules that meet minimum confidence level will go further to build up two-consequent rules as candidate rules. If those two-consequent rules meet the minimum confidence level will continue to build up three-consequent rules, and so on.

Even after limiting the possible rules to those that meet minimum support and confidence there is usually a very large number of rules generated, most of which are not valuable. Determining how to select the most important set of association rules is therefore an important issue in association rule discovery and here optimization can be applied. This issue has received moderate consideration in the recent years. Most of this research focuses on using the two metrics of support and confidence of association rules. The idea of optimized association rules was first introduced by Fukuda et al. (1996). In this work, an association rule R is of the form $(A \in [v_1, v_2]) \wedge C_1 \Rightarrow C_2$, where, A is a numeric attribute, v_1 and v_2 are a range of attribute A, and C_1 and C_2 are two normal attributes. Then the optimized association rules problem can be divided into two sub-problems.

On the other hand, the support of the antecedent can be maximized subject to the confidence of the rule R meeting the minimum confidence. Thus, the optimized support rule can be formulated as follows:

$$\begin{aligned} \max_R \quad & \sup\{(A_1 \in [v_1, v_2]) \wedge C_1\} \\ \text{s.t.} \quad & \mathrm{conf}(R) \geqslant \mathrm{conf}_{\min}. \end{aligned} \tag{18}$$

Alternatively, the confidence of the rule R can be maximized subject to the support of the antecedent being greater than the minimum support. Thus, the optimized confidence rule can be formulated as follows:

$$\begin{aligned} \max_R \quad & \mathrm{conf}\{(A_1 \in [v_1, v_2]) \wedge C_1\} \\ \text{s.t.} \quad & \sup(R) \geqslant \sup_{\min}. \end{aligned} \tag{19}$$

Rastogi and Shim (1998) presented optimized association rules problem to allow an arbitrary number of uninstantiated categorical and numeric attributes. Moreover, the authors proposed to use branch and bound and graph search pruning techniques to reduce the search space. In later work Rastogi and Shim (1999) extended this work to optimized support association rule problem for numeric attributes by allowing rules to contain disjunctions of uninstantiated numeric attributes. Dynamic programming is used to generate the optimized support rules and a bucketing technique and divide and conquer strategy are employed to improve the algorithm's efficiency.

By combining concerns with both support and confidence, Brin et al. (2000) proposed optimizing the gain of the rule, where gain is defined as

$$\begin{aligned} \mathrm{gain}(R) &= \sup\{(A_1 \\ &\in [v_1, v_2]) \wedge C_1\} - \mathrm{Conf}_{\min} \cdot \sup(C_1) \\ &= \sup(R) \cdot (\mathrm{Conf}(R) - \mathrm{Conf}_{\min}). \end{aligned} \tag{20}$$

The optimization problem maximizes the gain subject to minimum support and confidence.

$$\begin{aligned} \max \quad & \mathrm{gain}(R) \\ \text{s.t.} \quad & \sup(R) \geqslant \sup_{\min} \\ \mathrm{Conf}(R) \geqslant \mathrm{conf}_{\min}. \end{aligned} \tag{21}$$

Although there exists considerable research on the issue of generating good association rules, support and confidence are not always sufficient measures for identifying insightful and interesting rules. Maximizing support may lead to finding many trivial rules that not valuable for decision making. On the other hand, maximizing confidence may lead to finding too specific rules that are also not useful for decision making. Thus, relevant good measures and factors for identifying good association rules need more exploration. In addition, how to optimize the rules that have been obtained is an interesting and challenging issue that has not received much attention. Both of these areas are issue where the OR community can contribute in a significant way to the field of association rule discovery.

4. Applications in the management of electronic services

Many of the most important applications of data mining come in areas related to management of electronic services. In such applications, automatic data collection and storage is often cheap and straightforward, which generates large databases to which data mining can be applied. In this section we consider two such applications areas, namely customer relationship management and personalization. We choose these applications because of their importance and the usefulness of data mining for their solution, as well as for the relatively unexplored potential there is for incorporating optimization technology to enable and explain the data mining results.

4.1. Customer relationship management

Relationship marketing and customer relationship management (CRM) in general have become central business issues. With more intense competition in many mature markets companies have realized that development of relationship with more profitable customer is a critical factor to staying in the market. Thus, CRM techniques have been developed that afford new opportunities for businesses to act well in a relationship market. The focus of CRM is on the customer and the potential for increasing revenue, and in doing so it enhances the ability of a firm to compete and to retain key customers.

The relationship between a business and customers can be described as follows. A customer purchases products and services, while business is to market, sell, provide and service customers. Generally, there are three ways for business to increase the value of customers:

- increase their usage (or purchases) on the products or service that customers already have;
- sell customers more or higher-profitable products;
- keep customers for a longer time.

A valuable customer is usually not static and the relationship evolves and changes over time. Thus, understanding this relationship is a crucial part of CRM. This can be achieved by analyzing the customer life-cycle, or customer lifetime, which refers to various stages of the relationship between customer and business. A typical customer life-cycle is shown in Fig. 3.

First, acquisition campaigns are marketing campaigns that are directed to the target market and seek to interest prospects in a company's product or service. If prospects respond to company's

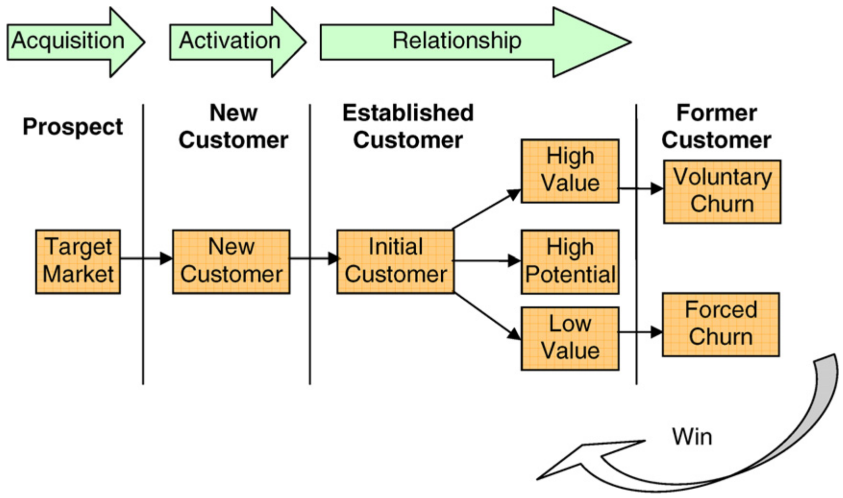

Fig. 3. Illustration of a customer life-cycle. This figure is adapted from Berry and Linoff (2000).

inquiry then they will become respondents. Responders become established customers when the relationship between them and the companies has been established. For example, they have made the initial purchase or their application for a certain credit card has been approved. At this point, companies will gain revenue from customer usage. Furthermore, customers' value will be increased not only by cross-selling that encourages customers to buy more products or services but also by up-selling that encourage customers to upgrading existing products and services. On the other hand, at some point established customers stop being customers (churn). There are two different types of churns. The first is voluntary churn, which means that established customers choose to stop being customers. The other type is forced churn, which refers to those established customers who no longer are good customers and the company cancels the relationship. The main purpose of CRM is to maximize customers' values throughout the life-cycle.

With large volumes of data generated in CRM data mining plays a leading role in the overall CRM (Rud and Brohman, 2003; Shaw et al., 2001). In acquisition campaigns data mining can be used to profile people who have responded to previous similar campaigns and these data mining profiles is helpful to find the best customer segments that the company should target (Adomavicius and Tuzhilin, 2003). Another application is to look for prospects who have similar behavior patterns to today's established customers. In responding campaigns data mining can be applied to determine which prospects will become responders and which responders will become established customers. Established customers are also a significant area for data mining. Identifying customer behavior patterns from customer usage data and predicting which customers are likely to respond to cross-sell and up-sell campaigns, which is very important to the business (Chiang and Lin, 2000). Regarding former customers, data mining can be used to analyze the reasons for churns and to predict churn (Chiang et al., 2003).

Optimization also plays an important role in CRM and in particular in determining how to develop proactive customer interaction strategy to maximize customer lifetime value. A customer is profitable if the revenue from this customer exceeds company's cost to attract, sell and service this customer. This excess is called the customer lifetime value (LTV). In other words, LTV is the total value to be gained while the customer is still active and it is one of the most important metric in CRM. There is much research that has been done in the area of modeling LTV using OR techniques (Schmittlein et al., 1987; Blattberg and Deighton, 1991; Dreze and Bonfrer, 2002; Ching et al., 2004).

Even when the LTV model can be formulated, it is difficult to find the optimal solution in the presence of great volume of data. Some researchers have addressed this by using data mining to find out the optimal parameters for the LTV model. For example, Rosset et al. (2002) formulate the following LTV model

$$\mathrm{LTV} = \int_0^\infty S(t)v(t)D(t)\,\mathrm{d}t, \tag{22}$$

where $v(t)$ describes the customer's value over time, $S(t)$ describes the probability that the customer is still active at time t, and $D(t)$ is a discounting factor. Data mining is then employed to estimate customer future revenue value and estimate the customer's churn probability over time from current data. This problem is difficult in practice, however, due to the large volumes of data involved. Padmanabhan and Tuzhilin (2003) presented two directions to reduce the complexity of the LTV optimization problem. One direction is to find good heuristics to improve LTV values and the other strategy is to optimize some simpler performance measures that are related to LTV value. As for the latter direction, the author pointed out that data mining and optimization can be integrated to build customer profiles, which is critical in many CRM applications. Data mining is first used for discover customer usage patterns and rules and optimization is then employed to select a small number of best patterns from the previously discovered rules. Finally, according to the customer profile, the company can achieve targeting and spend money on those customers who are likely to respond within their budget.

Campaign optimization is another problem where a combination of data mining and operation research can be applied. In the campaign optimization process a company needs to determine which kind of offers should go to which segment of customers or prospects through which communication channel. Vercellis (2002) presents two stages of campaign optimization models with both data mining technology and an optimization strategy. In the first stage optimization sub-problems are solved for each campaign and customers are segmented by their scores. In the second stage, a mixed integer optimization model is formulated to solve the overall campaign optimization problem based on customer segmentation and the limited available resources.

As described above, numerous CRM problems can be formulated into optimization problems but there are usually very large volumes of data that may make the problem difficult to solve. Combining the optimization problem with data mining is valuable in this context. For example, data mining can be used to identify insightful patterns from customer data and then these patterns can be used to identify more relevant constraints for the optimization models. In addition, data mining can be applied to reduce the search space and improve the computing time. Thus, investigating how to combine optimization and data mining to address CRM problems is a promising research area for the operation research community.

4.2. Personalization

Personalization is the ability to provide content and services tailored to individuals on the basis of knowledge about their preferences and behavior. Data mining research related to personalization has focused mostly on recommender systems and related issues such as collaborative filtering, and recommender systems have been investigated intensively in the data mining community (Breese et al., 1998; Geyer-Schulz and Hahsler, 2002; Lieberman, 1997; Lin et al., 2000). Such systems can be categorized into three groups: content-based systems, social data mining, and collaborative filtering. Content-based systems use exclusively the preferences of the user receiving the recommendation (Hill et al., 1995). These preferences are learned through implicit or explicit user feedback and typically represented as a profile for the user. Recommenders based on social data mining consider data sources created by groups of people as part of their daily activities and mine this data for potentially useful information. However, the recommendation of social data mining systems are usually not personalized but rather broadcast to the entire user community. On the other hand, such personalization is achieved by collaborative filtering (Resnick et al., 1994; Shardanan and Maes, 1995; Good et al., 1999), which matches users with similar interests and uses the preferences of these users to make recommendations.

As argued in Adomavicius and Tuzhilin (2003), the recommendation problem can be formulated as an optimization problem that selects the best items to recommend to a user. Specifically, given a set U of users and a set V of items, a ratings function $f: U \times V \to \mathbf{R}$ can be defined to specify how each user $u \in U$ likes each item $v \in V$. The recommendation problem can then be formulated as the following optimization problem:

$$\begin{aligned} \max \quad & f(u, v) \\ \text{subject to} \quad & u \in U \\ & v \in V. \end{aligned} \tag{23}$$

The challenge for this problem is that the rating function can usually only be partially specified, that is, not all entries in the matrix $\{f(u, v)\}_{u \in U, v \in V}$ have known ratings. Therefore, it is necessary to specify how unknown ratings should be estimated from the set of the previously specified ratings (Padmanabhan and Tuzhilin, 2003). Numerous methods have been developed for estimating these ratings and Pazzani (1999) and Adomavicius and Tuzhilin (2003) describe some of these methods. Once the optimization problem is defined data mining can contribute to its solution by learning additional constraints with data mining methods. For more on OR in personalization we refer the reader to Murthi and Sarkar (2003), but simply note that this area has a wealth of opportunities for the OR community to contribute.

5. Conclusions

As illustrated in this paper, the OR community has over the past several years made highly significant contributions to the growing field of data mining. The existing contributions of optimization methods in data mining touch on almost every part of the data mining process, from data visualization and preprocessing, to inductive learning, and selecting the best model after learning. Furthermore, data mining can be helpful in many OR application areas and can be used in a complementary way to optimization method to identify constraints and reduce the search space.

Although large volume of work already exists covering the intersection of OR and data mining we feel that the current work is only the beginning. Interest in data mining continues to grow in both academia and industry and most data mining issues where there is the potential to use optimization methods still require significantly more research. This is clearly being addressed at the present time, as interest in data mining within the OR community is growing, and we hope that this survey helps to further motivate more researchers to contribute to this exciting field.

References

Abbiw-Jackson, R., Golden, B., Raghavan, S., Wasil, E., 2006. A divide-and-conquer local search heuristic for data visualization. Computers and Operations Research 33, 3070–3087.

Adomavicius, G., Tuzhilin, A., 2003. Recommendation technologies: Survey of current methods and possible extensions. Working paper, Stern School of Business, New York University, New York.

Agrawal, R., Imielinski, T., Swami, A., 1993. Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 207–216.

Agrawal, R., Srikant, R., 1994. Fast algorithms for mining association rules. In: Proceedings of the 1994 International Conference on Very Large Data Bases (VLDB'94), pp. 487–499.

Bennett, K.P., 1992. Decision tree construction via linear programming. In: Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference, Utica, IL, pp. 97–101.

Bennett, K.P., Bredensteiner, E., 1999. Multicategory classification by support vector machines. Computational Optimizations and Applications 12, 53–79.

Bennett, K.P., Campbell, C., 2000. Support vector machines: Hype or hallelujah?. SIGKDD Explorations 2 (2) 1–13.

Berry, M., Linoff, G., 2000. Mastering Data Mining. Wiley, New York.

Blattberg, R., Deighton, J., 1991. Interactive marketing: Exploiting the age of addressability. Sloan Management Review, 5–14.

Borg, I., Groenen, P., 1997. Modern Multidimensional Scaling: Theory and Applications. Springer, New York.

Boros, E., Hammer, P.L., Ibaraki, T., Kogan, A., Mayoraz, E., Muchnik, I., 2000. Implementation of logical analysis of data. IEEE Transactions on Knowledge and Data Engineering 12 (2), 292–306.

Bradley, P.S., Fayyad, U.M., Mangasarian, O.L., 1999. Mathematical programming for data mining: Formulations and challenges. INFORMS Journal on Computing 11, 217–238.

Bradley, P.S., Fayyad, U.M., Reina, C., 1998a. Scaling clustering algorithms to large databases. In: Proceedings of ACM Conference on Knowledge Discovery in Databases, pp. 9–15.

Bradley, P.S., Mangasarian, O.L., 2000. *k*-Plane clustering. Journal of Global Optimization 16 (1), 23–32.

Bradley, P.S., Mangasarian, O.L., Street, N., 1996. Clustering via concave minimizationAdvances in Neural Information Processing Systems, vol. 9. MIT Press, Cambridge, MA, pp. 368–374.

Bradley, P.S., Mangasarian, O.L., Street, W.N., 1998b. Feature selection via mathematical programming. INFORMS Journal on Computing 10 (2), 209–217.

Breese, J., Heckerman, D., Kadie, C., 1998. Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Wadsworth International Group, Monterey, CA.

Brin, S., Rastogi, R., Shim, K., 2000. Mining optimized gain rules for numeric attributes. IEEE Transaction on Knowledge and Data Engineering 15 (2), 324–338.

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

Castillo, E., Gutiérrez, J.M., Hadi, A.S., 1997. Expert Systems and Probabilistic Network Models. Springer, Berlin.

Chiang, I., Lin, T., 2000. Using rough sets to build-up web-based one to one customer services. IEEE Transactions.

Chiang, D., Lin, C., Lee, S., 2003. Customer relationship management for network banking churn analysis. In: Proceedings of the International Conference on Information and Knowledge Engineering, Las Vegas, NV, 135–141.

Ching, W., Wong, K., Altman, E., 2004. Customer lifetime value: Stochastic optimization approach. Journal of the Operational Research Society 55 (8), 860–868.

Cooper, G.F., Herskovits, E., 1992. A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9, 309–347.

Cortes, C., Vapnik, V., 1995. Support vector networks. Machine Learning 20, 273–297.

Dasgupta, S., 2002. Performance guarantees for hierarchical clustering. In: Proceedings of the 15th Annual Conference on Computational Learning Theory, pp. 351–363.

Debuse, J.C., Rayward-Smith, V.J., 1997. Feature subset selection within a simulated annealing data mining algorithm. Journal of Intelligent Information Systems 9, 57–81.

Debuse, J.C., Rayward-Smith, V.J., 1999. Discretisation of continuous commercial database features for a simulated annealing data mining algorithm. Applied Intelligence 11, 285–295.

Dhar, V., Chou, D., Provost, F., 2000. Discovering interesting patterns for investment decision making with GLOWER – a genetic learner overlaid with entropy reduction. Data Mining and Knowledge Discovery 4, 251–280.

Dreze, X., Bonfrer, A., 2002. To pester or leave alone: Lifetime value maximization through optimal communication timing. Working paper, Marketing Department, University of California, Los Angeles, CA.

Estevill-Castro, V., 2002. Why so many clustering algorithms – a position paper. SIGKDD Explorations 4 (1), 65–75.

Fayyad, U., Piatetsky-Shapiro, G., Smith, P., Uthurusamy, R., 1996. Advances in Knowledge Discovery and Data Mining. MIT Press, Cambridge, MA.

Felici, G., Truemper, K., 2002. A MINSAT approach for learning in logic domains. INFORMS Journal on Computing 14, 20–36.

Folino, G., Pizzuti, C., Spezzano, G., 2001. Parallel genetic programming for decision tree induction. Tools with Artificial Intelligence. In: Proceedings of the 13th International Conference, pp. 129–135.

Freed, N., Glover, F., 1986. Evaluating alternative linear programming models to solve the two-group discriminant problem. Decision Sciences 17, 151–162.

Friedman, N., Geiger, D., Goldszmidt, M., 1997. Bayesian network classifiers. Machine Learning 29, 131–163.

Fu, Z., Golden, B., Lele, S., Raghavan, S., Wasil, E., 2003a. A genetic algorithm-based approach for building accurate decision trees. INFORMS Journal of Computing 15, 3–22.

Fu, Z., Golden, B., Lele, S., Raghavan, S., Wasil, E., 2003b. Genetically engineered decision trees: Population diversity produces smarter trees. Operations Research 51 (6), 894–907.

Fu, Z., Golden, B., Lele, S., Raghavan, S., Wasil, E., 2006. Diversification for smarter trees. Computers and Operations Research 33, 3185–3202.

Fukuda, T., Morimoto, Y., Morishita, S., Tokuyama, T., 1996. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In: Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 13–23.

Geyer-Schulz, A. Hahsler, M., 2002. Evaluation of recommender algorithms for and internet information broker based on simple association rules and the repeat-buying theory. In: Proceedings of WEBKDD'02, pp. 100–114.

Glover, 1990. Improved linear programming models for discriminant analysis. Decision Sciences 21 (4), 771–785.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic, Boston.

Glover, F., Laguna, M., Marti, R., 2003. Scatter search. In: Tsutsui, Ghosh (Eds.), Theory and Applications of Evolutionary Computation: Recent Trends. Springer, Berlin, pp. 519–528.

Glover, F., Kochenberger, G.A., 2003. Handbook of Metaheuristics. Kluwer Academic Publishers, Boston, MA.

Goldberg, D.E., 1989. Genetic Algorithm in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B., 1999. Combining collaborative filtering with personal agents for better recommendations. In: Proceedings of the National Conference on Artificial Intelligence.

Grabmeier, J., Rudolph, A., 2002. Techniques of cluster algorithms in data mining. Data Mining and Knowledge Discovery 6, 303–360.

Hansen, P., Mladenovic, N., 1997. An introduction to variable neighborhood search. In: Voss, S., et al. (Eds.), Proceedings of MIC 97 Conference.

Heckerman, D., Geiger, D., Chickering, D.M., 1995. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning 20 (3), 197–243.

Hill, W.C., Stead, L., Rosenstein, M., Furnas, G., 1995. Recommending and evaluating choices in a virtual community of use. In: Proceedings of the CHI'95 Conference on Human Factors in Computing Systems, pp. 194–201.

Hochbaum, D., Shmoys, D., 1985. A best possible heuristic for the k-center problem. Mathematics of Operations Research 10, 180–184.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press.

Hwang, F., 1981. Optimal partitions. Journal of Optimization Theory and Applications 34, 1–10.

Jain, A.K., Murty, M.N., Flynn, P.J., 1999. Data clustering: A review. ACM Computing Surveys 31, 264–323.

Jensen, F.V., 1996. An Introduction to Bayesian Networks. UCL Press Limited, London.

Joseph, A., Bryson, N., 1997. *W*-efficient partitions and the solution of the sequential clustering problem. Annals of Operations Research: Nontraditional Approaches to Statistical Classification 74, 305–319.

Kaufman, L., Rousseeuw, P.J., 1990. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

Kennedy, H., Chinniah, C., Bradbeer, P., Morss, L., 1997. The construction and evaluation of decision trees: A comparison of evolutionary and concept learning methods. In: Come, D., Shapiro, J. (Eds.), Evolutionary Computing, Lecture Notes in Computer Science. Springer, Berlin, pp. 147–161.

Kim, J., Olafsson, S., 2004. Optimization-based data clustering using the nested partitions method. Working paper, Department of Industrial and Manufacturing Systems Engineering, Iowa State University.

Kim, Y., Street, W.N., Menczer, F., 2000. Feature selection in unsupervised learning via evolutionary search. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00), pp. 365–369.

Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Kroese, D.P, Rubinstein, R.Y., Taimre, T., 2004. Application of the Cross-Entropy Method to Clustering and Vector Quantization. Working Paper.

Lam, W., Bacchus, F., 1994. Learning Bayesian belief networks. An approach based on the MDL principle. Computational Intelligence 10, 269–293.

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R., Kuijpers, C., 1996. Structure learning of bayesian network by genetic algorithms: A performance analysis of control parameters. IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (9), 912–926.

Lauritzen, S.L., 1995. The EM algorithm for graphical association models with missing data. Computational Statistics and Data Analysis 19, 191–201.

Lee, J.-Y., Olafsson, S., 2006. Multiattribute decision trees and decision rules. In: Triantaphyllou, Felici (Eds.), Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques, pp. 327–358.

Li, X., Olafsson, S., 2005. Discovering dispatching rules using data mining. Journal of Scheduling 8 (6), 515–527.

Lieberman, H., 1997. Autonomous interface agents. In: Proceedings of the CHI'97 Conference on Human Factors in Computing Systems, pp. 67–74.

Lin, W., Alvarez, S.A., Ruiz, C., 2000. Collaborative recommendation via adaptive association rule mining. In: Proceedings of ACM WEBKDD 2000.

Liu, H., Motoda, H., 1998. Feature Selection for Knowledge Discovery and Data Mining. Kluwer, Boston.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297.

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

Mangasarian, O.L., 1994. Misclassification minimization. Journal of Global Optimization 5 (4), 309–323.

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

Murthi, B.S., Sarkar, Sumit, 2003. The role of the management sciences in research on personalization. Management Science 49 (1), 1344–1362.

Narendra, P.M., Fukunaga, K., 1977. A branch and bound algorithm for feature subset selection. IEEE Transactions on Computers 26 (9), 917–922.

Ng, R., Han, J., 1994. Efficient and effective clustering method for spatial data mining. In: Proceedings for the 1994 International Conference on Very Large Data Bases, pp. 144–155.

Niimi, A., Tazaki, E., 2000. Genetic programming combined with association rule algorithm for decision tree construction. In: Fourth International Conference on Knowledge-based Intelligent Engineering Systems and Allied Technologies, Brighton, UK, pp. 746–749.

Olafsson, S., Yang, J., 2004. Intelligent partitioning for feature selection. INFORMS Journal on Computing 17 (3), 339–355.

Osei-Bryson, K.-M., 2005. Assessing cluster quality using multiple measures – a decision tree based approach. In: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Next Wave in Computing, Optimization, and Decision Technologies. Kluwer.

Padmanabhan, B., Tuzhilin, A., 2003. On the use of optimization for data mining: Theoretical interactions and eCRM opportunities. Management Science 49 (10), 1327–1343.

Pazzani, M., 1999. A framework for collaborative, content-based and demographic filtering. Artificial Intelligence Review, 393–408.

Pernkopf, F., 2005. Bayesian network classifiers versus selective *k*-NN classifier. Pattern Recognition 38 (1), 1–10.

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan-Kaufmann, San Mateo, CA.

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

Rastogi, R., Shim, K., 1998. Mining optimized association rules for categorical and numeric attributes. In: Proceedings of International Conference of Data Engineering.

Rastogi, R., Shim, K., 1999. Mining optimized association support rules for numeric attributes. In: Proceedings of International Conference of Data Engineering.

Resnick, P., Iacovou, N., Suckak, M., Bergstrom, P, Riedl, J., 1994. Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work, pp. 175–186.

Resende, M.G.C., Ribeiro, C.C., 2003. Greedy randomized adaptive search procedures. In: Kochenberger, Glover (Eds.), Handbook of Metaheuristics. Kluwer, Boston, MA.

Ripley, B.D., 1996. Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge, UK.

Rosset, S., Neumann, E., Vatnik, Y., 2002. Customer lifetime value modeling and its use for customer retention planning. In: Proceedings of ACM International Conference on Knowledge Discovery and Data mining.

Rumelhart, D.E., McClelland, J.L., 1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition 1. MIT Press, Cambridge, MA.

Schmittlein, D., Morrison, D., Colombo, R., 1987. Counting your customers: Who are they and what will they do next?. Management Science 33 (1) 1–24.

Shardanan, U., Maes, P., 1995. Social information filtering: Algorithms for automating 'word of mouth'. In: Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems, pp. 210–17.

Sharpe, P.K., Glover, R.P., 1999. Efficient GA based technique for classification. Applied Intelligence 11, 277–284.

Shaw, M., Subramaniam, C., Tan, G., Welge, M., 2001. Knowledge management and data mining for marketing. Decision Support Systems 31, 127–137.

Shi, L., Olafsson, S., 2000. Nested partitions method for global optimization. Operations Research 48, 390–407.

Shmoys, D.B., 1999. Approximation algorithms for clustering problems. In: Proceedings of the 12th Annual Conference on Computational Learning Theory, pp. 100–101.

Shmoys, D.B, Tardos, É., Aardal, K., 1997. Approximation algorithms for facility location problems. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, pp. 265–274.

Street, W.N., 2005. Multicategory decision trees using nonlinear programming. Informs Journal on Computin 17, 25–31.

Suzuki, J., 1993. A construction of Bayesian networks from databases based on an MDL scheme. In: Heckerman, D., Mamdani, A. (Eds.), Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, pp. 266–273.

Tanigawa, T., Zhao, Q.F., 2000. A study on efficient generation of decision trees using genetic programming. In: Proceedings of the 2000 Genetic and Evolutionary Computation Conference (GECCO'2000), Las Vegas, pp. 1047–1052.

Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer, Berlin.

Vapnik, V., Lerner, A., 1963. Pattern recognition using generalized portrait method. Automation and Remote Control 24, 774–780.

Vercellis, C., 2002. Combining data mining and optimization for campaign management. Management Information SystemsData Mining III, vol. 6. Wit Press, pp. 61–71.

Vinod, H.D., 1969. Integer programming and the theory of grouping. Journal of the American Statistical Association 64, 506–519.

Weiss, S.M., Kulikowski, C.A., 1991. Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems. Morgan Kaufman.

Yang, J., Honavar, V., 1998. Feature subset selection using a genetic algorithm. In: Motada, H., Liu, H. (Eds.), Feature Selection, Construction, and Subset Selection: A Data Mining Perspective. Kluwer, New York.

Yang, J., Olafsson, S., 2006. Optimization-based feature selection with adaptive instance sampling. Computers and Operations Research 33, 3088–3106.

Zhang, G.P., 2000. Neural networks for classification: A survey. IEEE Transactions on Systems Man and Cybernetics Part C – Applications and Reviews 30 (4), 451–461.

Zhang, T., Ramakrishnan, R., Livny, M., 1996. BIRCH: An efficient data clustering method for very large databases. In: SIGMOD Conference, pp. 103–114.