Data mining: machine learning, statistics, and databases

Heikki Mannila*
Department of Computer Science
University of Helsinki,
FIN-00014 Helsinki, Finland
E-mail: `Heikki.Mannila@cs.helsinki.fi`
URL: `http://www.cs.helsinki.fi/~mannila/`

Abstract

Knowledge discovery in databases and *data mining* aim at semiautomatic tools for the analysis of large data sets. We give an overview of the area and present some of the research issues, especially from the database angle.

1 Introduction

Knowledge discovery in databases (KDD), often called data mining, aims at the discovery of useful information from large collections of data. The discovered knowledge can be rules describing properties of the data, frequently occurring patterns, clusterings of the objects in the database, etc. Data mining has in the 1990's emerged as visible research and development area; both in industry and in science there seems to be a lack of methods for efficient analysis of large data sets.

Current technology makes it fairly easy to collect data, but data analysis tends to be slow and expensive. There is a suspicion that there might be nuggets of useful information hiding in the masses of unanalyzed or underanalyzed data, and therefore semiautomatic methods for locating interesting information from data would be useful. There are several successful applications of data mining. See [9] for a recent overview of the area.

This paper gives a short discussion of some of the database-oriented research issues in knowledge discovery. We start in Section 2 by briefly discussing the KDD process, basic data mining techniques, and listing some prominent applications. Section 3 discusses the role of machine learning and statistics in KDD and data mining. In Section 4 we move to the role of databases in knowledge discovery by looking at a simple example of data mining, namely the problem of discovering association rules. We present a simple algorithm for this task, and show in Section 5 how the same ideas can be used for other types of data, too. Section 6 moves to a generic data mining algorithm, and discusses some of the architectural issues in data mining systems. Based on these ideas, we consider in Section 7 the possibilities of specifying KDD tasks in a high level language and compiling these specifications to efficient discovery algorithms. Section 8 considers the possibilities of representing large data sets by smaller condensed representations, and Section 9 is a short conclusion.

2 The KDD process

The goal of knowledge discovery is to obtain useful knowledge from large collections of data. Such a task is inherently interactive and iterative: one cannot expect to obtain useful knowledge simply by pushing a lot of data to a black box. The user of a KDD system has to have a solid understanding of the domain in order to select the right subsets of data, suitable classes of patterns, and good criteria for interestingness of the patterns. Thus KDD systems should be seen as a interactive tools, not as automatic analysis systems.

Discovering knowledge from data should therefore be seen as a process containing several steps:

1. understanding the domain;
2. preparing the data set;
3. discovering patterns (data mining),
4. postprocessing of discovered patterns, and
5. putting the results into use.

(See [8] for a slightly different process model and excellent discussion.)

*Part of this work was done while the author was visiting the Max Planck Institut für Informatik in Saarbrücken, Germany. Work supported by the Academy of Finland and by the Alexander von Humboldt Stiftung.

Understanding the domain of the data is naturally a prerequisite for extracting anything useful: the user of a KDD system has to have some sort of understanding about the application area before any valuable information can be obtained. On the other hand, if very good human experts exist for a domain, it can be hard for semiautomatic tools to obtain any novel information. This can be the case in fairly stable domains, where the humans have had time to achieve expertise even in the details of the data. A possible example occurs in some areas of retailing, where the products and customer profiles can stay about the same for longer periods of time. The easiest application areas for KDD seem to be ones where general human experts can be found, but the actual microlevel properties of the data are changing. This seems to be the case in, e.g., telecommunications, where the operators of the networks have a fairly good overview of the systems characteristics, but changes and updates in equipment and software mean that actual expertise in the details of the data is more difficult to obtain.

Preparation of the data set involves selection of the data sources, integration of heterogeneous data, cleaning the data from errors, assessing noise, dealing with missing values, etc. This step can easily take up to 80 % of the time needed for the whole KDD process; this is not surprising, since the difficulties in data integration are well known.

The pattern discovery phase in KDD is the step where the interesting and frequently occurring patterns are discovered from the data. In this paper we follow the terminology introduced in [8]: data mining refers to the pattern discovery part of knowledge discovery. Elsewhere, especially in industry, data mining is often used as a synonym for KDD.

The data mining step can use various techniques from statistics and machine learning, such as rule learning, decision tree induction, clustering, inductive logic programming, etc. The emphasis in data mining research is mostly on efficient discovery of fairly simple patterns.

The KDD process does not stop when patterns have been discovered. The user has to be able to understand what has been discovered, to view the data and patterns simultaneously, contrast the discovered patterns with background knowledge, etc. Postprocessing of discovered knowledge involves steps such as further selection or ordering of patterns, visualization, etc. Some approaches to KDD methodology put a heavy emphasis on postprocessing.

The KDD process is necessarily iterative: the results of a data mining step can show that some changes should be made to the data set formation step, postprocessing of patterns can cause the user to look for some slightly modified types of patterns, etc. Efficient support for such iteration is one important development topic in KDD.

Prominent applications of KDD include health care data, financial applications, and scientific data [26, 19]. One of the more spectacular applications is the SKICAT system [7], which operates on 3 terabytes of image data, producing a classification of approximately 2 billion sky objects into a few classes. The task is obviously impossible to do manually. Using example classifications provided by the users, the system learns classification rules that are able to do the categorization accurately and fast.

In industry, the success of KDD is partly related to the rise of the concepts of data warehousing and on-line analytical processing (OLAP). These strategies for the storage and processing of the accumulated data in an organization have become popular in recent years. KDD and data mining can be viewed as ways of realizing some of the goals of data warehousing and OLAP.

3 Data mining, machine learning, and statistics

Data mining combines methods and tools from at least three areas: machine learning, statistics, and databases. Indeed, one can sometimes hear the following comments.

- Data mining is just machine learning!
- Data mining is just statistics!
- What does data mining have to do with databases?

In this section we discuss briefly the first two points; the rest of the paper is devoted to a discussion on the third point.

The close links between machine learning, statistics, and data mining are fairly obvious. All three areas aim at locating interesting regularities, patterns, or concepts from empirical data. The exact relationships of these areas have been subject to some debate.

Machine learning methods form the core of data mining: decision tree learning or rule induction is one of the main components of several data mining algorithms. There are also some differences, however.

The emphasis on the process of knowledge discovery is one; large parts of machine learning literature concentrate on just the learning or induction step, although exceptions of course exist.

The next difference concerns the relative roles of concepts and data. It seems to me that most machine learning research assumes there is *something to be learned*, i.e., that there is an underlying interesting concept or mechanism that produces the data. The data can be corrupted by noise, errors, etc., but still the idea is that there is an interesting concept at the bottom. In knowledge discovery, on the other hand, the data is the primary thing, and one does not necessarily assume that there would be any sensible structure behind the data. For example, in analyzing retail sales data, the data is what it is, and the users are not interested in obtaining a full understanding of it; useful nuggets of information are sufficient. Of course, this difference is not absolute.

A third difference is related to the goals. KDD systems typically have fairly modest aims, in terms of the complexity of the obtained knowledge. Whereas parts of machine learning research aim at learning things that are difficult for humans to do, most of the work in KDD aims at finding knowledge that a competent data analyst would in principle be able to find, if he/she had the time. This distinction is particularly evident when one compares the area of machine discovery to knowledge discovery.

An often cited difference between KDD and machine learning is the amount of data. Traditionally, machine learning research has concentrated on looking at data sets containing hundreds or thousands of examples, while KDD applications consider larger data sets. I'm not sure how significant this distinction is, however: some machine learning work has been done on huge data sets, and KDD methods can be a useful even on small data collections. Furthermore, the essential source of complexity in data mining is typically not the number of objects in the database, but rather the number of attributes: the number of possible patterns typically grows at least exponentially in the number of attributes. This growth is the real source of difficulty, not the number of objects in the database.

Summarizing, machine learning is at the core of KDD, but there are differences between the areas.

In *statistics* the term data mining has been used for a long time, often in slightly derogatory fashion, as referring to data analysis without clearly formulated hypotheses. A more fashionable term is *exploratory data analysis* (EDA) [29], which stressed the supremacy of data as guiding the analysis process. KDD and EDA have very similar aims and methods.

According to the interesting statistical perspective on KDD by Elder and Pregibon [6], the focus of statistics has gradually moved from model estimation to model selection. Instead of looking for the parameter values that make a model fit the data well, also the model structure is part of the search process. This trend fits the goals of KDD nicely: one does not want to fix the model structure in advance. The recent advances in, say Markov chain Monte Carlo (MCMC) methods, make it possible to consider far larger model spaces than previously. In addition to these techniques, the KDD community has lots to learn from statistics, e.g., in the handling of uncertainty.

The main difference between KDD and statistics is perhaps in the extensive use of machine learning methods in KDD, in the volume of data, and in the role of computational complexity issues in KDD. For example, even MCMC methods have difficulties in handling tens of thousands of parameter values; some sort of combinatorial preprocessing is needed to make the model selection task tractable. It seems that such combinations of methods can be useful: combinatorial techniques are used to prune the search space, and statistical methods are used to explore the remaining parts in great detail.

4 Databases and data mining

What is the role of databases in data mining? Database management systems are systems especially developed for the storage and flexible retrieval of large masses of structured data, so in principle they should be suited for KDD, including data mining.

In the remaining sections of this paper we discuss some issues that are connected with the use of database management systems in KDD applications. We introduce the issues by considering first a simple data mining problem, and then generalizing it.

The problem we consider is finding *association rules* from binary data. Assume we have a set $R = \{A_1, \ldots, A_p\}$ of binary attributes, i.e., the domain of each A_i is $\{0, 1\}$. A relation $r = \{t_1, \ldots, t_n\}$ over the schema R is a matrix with columns R and n rows, each row being a vector of length p of 0's and 1's.

An *association rule* [1] about r is an expression of the form $X \Rightarrow B$, where $X \subseteq R$ and $B \in R \setminus X$. The intuitive meaning of the rule is that if a row of the matrix r has a 1 in each column of X, then the row tends to have a 1 also in column B.

Examples of data where association rules might be applicable include the following.

- A student database at a university: rows correspond to students, columns to courses, and a 1 in entry (s, c) indicates that the student s has taken course c.

- Data collected from bar-code readers in supermarkets: columns correspond to products, and each row corresponds to the set of items purchased at one time.

- A database of publications: the rows and columns both correspond to publications, and $(p, p') = 1$ means that publication p refers to publication p'.

- A set of measurements about the behavior of a bunch of systems, say exchanges in a telephone network. The columns correspond to the presence or absence of certain conditions, and each row correspond to a measurement: if entry (m, c) is 1, then at measurement m condition c was present.

Given $W \subseteq R$, we denote by $s(W, r)$ the *frequency* of W in r: the fraction of rows of r that have a 1 in each column of W. The *frequency* of the rule $X \Rightarrow B$ in r is defined to be $s(X \cup \{B\}, r)$, and the *confidence* of the rule is $s(X \cup \{B\}, r)/s(X, r)$.

In discovery of association rules, the task is to find all rules $X \Rightarrow B$ such that the frequency of the rule is at least a given threshold σ and the confidence of the rule is at least another threshold θ. In large retailing applications the number of rows might be 10^6 or even 10^7, and the number of columns around 5000. The frequency threshold σ typically is around 10^{-2} — 10^{-5}. The confidence threshold θ can be anything from 0 to 1. From such a database one might obtain hundreds or thousands of association rules. (Of course, one has to be careful in assigning any statistical significance to findings obtained from such methods.)

Note that there is no predefined limit on the number of attributes of the left-hand side X of an association rule $X \Rightarrow B$; this is important so that unexpected associations are not ruled out before the processing starts. It also means that the search space of the rules has exponential size in the number of attributes of the input relation. Handling this requires some care for the algorithms, but there is a simple way of pruning the search space.

We call a subset $X \subseteq R$ *frequent* in r, if $s(X, r) \geq \sigma$. Once all frequent sets of r are known, finding the association rules is easy. Namely, for each frequent set X and each $B \in X$ verify whether the rule $X \setminus \{B\} \Rightarrow B$ has sufficiently high confidence.

How can one find all frequent sets X? This can be done in a multitude of ways [1, 2, 12, 14, 28]. A typical approach [1, 2] is to use that fact that all subsets of a frequent set are also frequent.

First find all frequent sets of size 1 by reading the data once and recording the number of times each attribute A occurs. Then form *candidate* sets of size 2 by taking all pairs $\{B, C\}$ of attributes such that $\{B\}$ and $\{C\}$ both are frequent. The frequency of the candidate sets is again evaluated against the database. Once frequent sets of size 2 are known, candidate sets of size 3 can be formed; these are sets $\{B, C, D\}$ such that $\{B, C\}$, $\{B, D\}$, and $\{C, D\}$ are all frequent. This process is continued until no more candidate sets can be formed.

As an algorithm, the process is as follows.

$\mathcal{C} := \{\{A\} \mid \mid A \in R\}$;
$\mathcal{F} := \emptyset$;
while $\mathcal{C} \neq \emptyset$ **do**
    $\mathcal{F}' :=$ the sets $X \in \mathcal{C}$ that are frequent;
    add $\mathcal{F}'$ to $\mathcal{F}$;
    $\mathcal{C} :=$ new candidate sets generated from $\mathcal{F}'$;
od;

The algorithm has to read the database at most $K+1$ times, where K is the size of the largest frequent set. In the applications, K is small, typically at most 10, so the number of passes through the data is reasonable. (Note that there are at least 2^K frequent sets; thus, if K is much larger than 10, the output of the frequent set algorithm gets quite big. There still might be only few interesting association rules, however.)

A modification of the above method is obtained by computing for each frequent set X the subrelation $r_X \subseteq r$ consisting of those rows $t \in r$ such that $t[A] = 1$ for all $A \in X$. Then it is easy to see that for example $r_{\{A,B,C\}} = r_{\{A,B\}} \cap r_{\{B,C\}}$. Thus the relation r_X for a set X of size k can be obtained from the relations $r_{X'}$ and $r_{X''}$, where $X' = X \setminus \{A\}$ and $X'' = X \setminus \{B\}$ for some $A, B \in X$ with $A \neq B$. This method has the advantage that rows that do not contribute to any frequent set will not be inspected more than once. For comparisons of the two approaches, see [2, 14, 28].

The algorithms described above work quite nicely on large input relations. Their running time is approximately $O(NF)$, where $N = np$ is the size of the input and F is the size of the output (the sum of the sizes of the frequent sets).[1] This is nearly linear, and the algorithms seem to scale nicely to tens of millions of examples. The only case when they fail is when the output is too large, i.e., there are too many frequent sets, but in that case the whole task is not very sensible.

The methods for finding frequent sets are simple: they are based on one nice but simple observation

[1] This is actually not quite true: the running time is $O(NF')$, where F' is the sum of the sizes of the frequent sets and the sets in the candidate collection $\mathcal{C}$ during the operation of the algorithm [24].

(subsets of frequent sets must be frequent), and use straightforward implementation techniques.

A naive implementation of the algorithms on top of a relational database system would be easy: we need to pose to the database management system queries of the form "What is $s(\{A_1, \ldots, A_k\}, r)$?", or in SQL

```
select count(*) from r t
where t[A_1] = 1 and ··· and t[A_k] = 1
```

The number of such queries can be large: if there are thousands of frequent sets, there will be thousands of queries. The overhead in performing the queries on an ordinary dbms would probably be prohibitive.

The customized algorithms described above are able to evaluate masses of such queries reasonably efficiently, for several reasons. First, all the queries are very simple, and have the same general form; thus there is no need to compile each query individually. Second, the algorithms that make repeated passes through the data evaluate a large collection of queries during a single pass. Third, the algorithm that build the relations r_X for frequent sets X use the results of previous queries to avoid looking at the whole data for each query.

5 Application: finding episodes from sequences

The basic ideas of the algorithm for finding association rules are fairly widely applicable. In this section we describe an application of the same ideas to the problem of finding repeated *episodes* in sequences of events.

Consider a sequence of events (e, t), where e is the type of the event and t is the time when the event occurred. Such data is routinely collected in, e.g., telecommunication networks, process monitoring, quality control, user interface studies, and epidemiology. There is an extensive statistical literature on how such data can be analyzed, but most methods are suited only for small numbers of event types.

For example, a telecommunications network alarm database is used to collect all the notifications about abnormal situations in the network. The number of event types is around 200, and there are 1000-10000 alarms per day [13].

As a first step in analyzing such data, one can try to find which event types occur frequently close together. Denoting by E the set of all event types, an *episode* φ is a partially ordered set of elements from E. An episode might, e.g., state that events of type A and B occur (in either order) before an event of type C.

Given an alarm sequence $r = (e_1, t_1), \ldots, (e_n, t_n)$, a *slice* r_t of r of width W consists of those events (e_i, t_i) of r such that $t \leq t_i \leq t + W$. An episode φ *occurs* in r_t, if there are events in r_t corresponding to the event types of φ and they occur in an order respecting the partial order of φ. An episode is *frequent*, if it occurs in sufficiently many slices of the original sequence.

How to find all episodes from a long sequence of events? Using the same idea as before, we first locate frequent episodes of size 1, then use these to generate candidate episodes of size 2, verify these against the data, generate candidate episodes of size 3, etc. The algorithm can be further improved by using incremental recognition of episodes; see [25] for details, and [22] for extensions. The results are good: the algorithms are efficient, and using them one can find easily comprehensible results about the combinations of event types that occur together.

6 A generic data mining algorithm and architecture

A fairly large class of data mining tasks can be described as the search for interesting and frequently occurring patterns from the data. That is, we are given a class $\mathcal{P}$ of patterns or sentences that describe properties of the data, and we can specify whether a pattern $p \in \mathcal{P}$ occurs frequently enough and is otherwise interesting. That is, the generic data mining task is to find the set

$$PI(d, \mathcal{P}) = \{p \in \mathcal{P} \mid p \text{ occurs sufficiently often in } d \text{ and } p \text{ is interesting}\}.$$

This point of view has either implicitly or explicitly been used in discovering integrity constraints from databases, in inductive logic programming, and in machine learning [4, 5, 18, 19, 20]; some theoretical results can be found in [24], and a suggested logical formalism in [17].

For association rules, the pattern class is the set of all rules of the form $X \Rightarrow B$, and a rule is interesting if its confidence is sufficiently high. For finding episodes, the patterns are the episodes and there need not be any interestingness criterion.

The algorithm given above can be generalized to finding $PI(d, \mathcal{P})$ as follows.

$\mathcal{C}$:= initial patterns from $\mathcal{P}$;
while $\mathcal{C} \neq \emptyset$ **do**
  for each $p \in \mathcal{C}$
    find the number of occurrences of p in d;
  $\mathcal{F} := \mathcal{F} \cup \{p \in \mathcal{C} \mid p$ is sufficiently frequent in $d\}$;
  $\mathcal{C}$:= new candidates generated from $\mathcal{F}$;
od;
output interesting patterns from $\mathcal{F}$;

In addition to discovering association rules or frequent episodes, istantiations of this algorithm include finding keys of relations [21], hill-climbing searches for best descriptions [15, 19], etc. In hill-climbing, the set $\mathcal{C}$ will contain only the neighbors of the current "most interesting" pattern. For other strategies, a possible problem with this algorithm is that the test for interestingness is applied only at the end. If there are lots of frequent but uninteresting patterns, the algorithm will use a lot of time inspecting these.

The generic algorithm suggests a data mining system architecture consisting of a discovery module and a database management system. The discovery module sends queries to the database, and the database answers. The queries are typically of the form "How many objects in the database match p", where p is a possibly interesting pattern; the database answers by giving the count.

If implemented naively, this architecture leads to slow operations. To achieve anything resembling the efficiency of tailored solutions, the database management system should be able to utilize the strong similarities between the queries generated by the discovery module.

The view of data mining as locating frequently occurring and interesting patterns from data suggests that data mining can benefit from the extensive research done in the area of *combinatorial pattern matching*; see, e.g., [10].

7 Towards higher-level data mining

Currently, data mining research and development consists mainly of isolated applications. Among others, Imielinski [16] has eloquently argued that data mining is today at the same state as data management was in the 1960's: then all data management applications were *ad hoc*; only the advent of the relational model and powerful query languages made it possible to develop applications fast. Consequently, data mining would need a similar theoretical framework.

The approach of computing $PI(d, \mathcal{P})$ presented in the previous section provides one possible framework. A data mining task could be given by specifying the class $\mathcal{P}$ and the interestingness predicate. That is, the user of a KDD system makes a *pattern query*. Mimicking the behavior of an ordinary dbms, the KDD system compiles and optimizes the pattern query, and executes it by searching for the patterns that satisfy the user's specifications and have sufficient frequency in the data.

As an example, consider the simple case of mining for association rules in a course enrollment database. The user might say that he/she is interested only in rules that have the "Data Management" course on the left-hand side. This restriction can be utilized in the algorithm for finding frequent sets: only candidate sets that contain "Data Management" need to be considered.

Developing methods for such pattern queries and their optimization is currently one of the most interesting research topics in data mining. So far, even the simple techniques such as the above have not been sufficiently studied. It is not clear how far one can get by using such methods, but the possible payoff is large.

In addition to developing query processing strategies for data mining applications, changes in the underlying storage model can also have a strong effect on the performance. A very interesting experiment in this direction is the work on the Monet database server developed at CWI in the Netherlands by Martin Kersten and others [15, 3]. The Monet system is based on the vertical partitioning of the relations: a relation with k attributes is decomposed into k relations, each with two attributes: the OID and one of the original attributes. The system is built on the extensive use of main memory, has an extensible set of basic operations, and supports shared-memory parallelism. Experiments with Monet on data mining applications have produced quite good results [15, 14].

8 Condensed representations

We remarked in Section 2 that KDD is an iterative process. Once a data mining algorithm has been used to discover potentially interesting patterns, the user often wants to view these patterns in different ways, have a look at the actual data, visualize the patterns, etc. A typical phenomenon is also that some pattern p looks interesting, and the user wants to evaluate other patterns that closely resemble p. In implementing such queries, caching of previous results is obviously useful. Still, having to go back to the original data each time the user wants some more information seems somewhat wastful. Similarly, in the generic data mining algorithm presented in Section 6 the frequency and interestingness of each pattern are verified against the database. It would be faster to look at some sort of short representation of the data.

Given a structure $d \in \mathcal{D}$, and a class of patterns $\mathcal{P}$, a *condensed representation* for d and $\mathcal{P}$ is a data structure that makes it possible to answer queries of the form "How many times does $p \in \mathcal{P}$ occur in d" approximately correctly and more efficiently than by looking at d itself.

A simple example of a condensed representation is obtained by taking a sample from the data: by counting the occurrences of the pattern in the sample, one gets an approximation of the number of occurrences in the original data. Another, less obvious example is given by the collection of frequent sets of a 0-1 valued relation [23]: the collection of frequent sets can be used to give approximate answers to arbitrary boolean queries about the data, even though the frequent sets represent only conjunctive concepts. The data cube [11] can also be viewed as a condensed representation for a class of queries. Similarly, in computational geometry the notion of an ε-approximation [27] is closely related.

Developing condensed representations for various classes of patterns seems a promising way of improving the effectiveness of data mining algorithms. Whether this approach is generally useful is still open.

9 Concluding remarks

Data mining and knowledge discovery are currently quite popular, and there is lots of hype around. As the area is partly a mixture of techniques from different fields, there is also the danger of reinventing old (bad) solutions.

We summarize some of the database related research directions.

1. Development of pattern query specification languages and techniques for optimizing pattern queries.
2. Condensed representations for various classes of patterns.
3. Caching strategies for processing strongly related queries.
4. Combinations of data mining and statistical techniques.
5. Using background knowledge (e.g., metadata) in the KDD process.
6. Tools for selecting, grouping, and visualizing discovered knowledge.

Acknowledgements

Discussions with Tomasz Imielinski, Willi Kloesgen, Arno Siebes, and Stefan Wrobel have been most useful. Pirjo Ronkainen and Hannu Toivonen provided comments on an earlier version.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207 – 216, May 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, Menlo Park, CA, 1996.

[3] P. A. Boncz, W. Quak, and M. L. Kersten. Monet and its geographical extensions: a novel approach to high-performance GIS processing. In *EDBT'96*, 1996. To appear.

[4] L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI–93)*, pages 1058 – 1053, Chambéry, France, 1993. Morgan Kaufmann.

[5] L. De Raedt and S. Džeroski. First-order *jk*-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375 – 392, 1994.

[6] J. Elder IV and D. Pregibon. A statistical perspective on knowledge discovery in databases. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 83 – 113. AAAI Press, Menlo Park, CA, 1996.

[7] U. M. Fayyad, S. G. Djorgovski, and N. Weir. Automating the analysis and cataloging of sky surveys. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 471 – 494. AAAI Press, Menlo Park, CA, 1996.

[8] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1 –34. AAAI Press, Menlo Park, CA, 1996.

[9] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.

[10] Z. Galil and E. Ukkonen, editors. *6th Annual Symposium on Combinatorial Pattern Matching (CPM 95) : Espoo, Finland, July 1995*, volume 937 of *Lecture Notes in Computer Science*, Berlin, 1995. Springer.

[11] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data Cube: A relational aggregation operator generalizing group-by, cross-tab, and subtotals. In *12th International Conference on Data Engineering (ICDE'96)*, pages 152 – 159, New Orleans, Louisiana, Feb. 1996.

[12] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 420 – 431, Zurich, Swizerland, 1995.

[13] K. Hätönen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen. Knowledge discovery from telecommunication network alarm databases. In *12th International Conference on Data Engineering (ICDE '96)*, pages 115 – 122, New Orleans, Louisiana, Feb. 1996.

[14] M. Holsheimer, M. Kersten, H. Mannila, and H. Toivonen. A perspective on databases and data mining. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 150 – 155, Montreal, Canada, Aug. 1995.

[15] M. Holsheimer, M. Kersten, and A. Siebes. Data surveyor: Searching the nuggets in parallel. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 447 – 467. AAAI Press, Menlo Park, CA, 1996.

[16] T. Imielinski. A database view on data mining. Invited talk at the KDD'95 conference.

[17] M. Jaeger, H. Mannila, and E. Weydert. Data mining as selective theory extraction in probabilistic logic. In *SIGMOD'96 Data Mining Workshop*, 1996. To appear.

[18] J.-U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. In S. Muggleton, editor, *Inductive Logic Programming*, pages 335 – 359. Academic Press, London, 1992.

[19] W. Kloesgen. Efficient discovery of interesting statements in databases. *Journal of Intelligent Information Systems*, 4(1):53 – 69, 1995.

[20] H. Mannila and K.-J. Räihä. Design by example: An application of Armstrong relations. *Journal of Computer and System Sciences*, 33(2):126 – 141, 1986.

[21] H. Mannila and K.-J. Räihä. *Design of Relational Databases*. Addison-Wesley Publishing Company, Wokingham, UK, 1992.

[22] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. Technical Report C-1996-12, University of Helsinki, Department of Computer Science, March 1996.

[23] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. Technical Report C-1996-13, University of Helsinki, Department of Computer Science, March 1996.

[24] H. Mannila and H. Toivonen. On an algorithm for finding all interesting sentences. In *Cybernetics and Systems Research '96*, Vienna, Austria, Apr. 1996. To appear.

[25] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210 – 215, Montreal, Canada, Aug. 1995.

[26] C. J. Matheus, G. Piatetsky-Shapiro, and D. McNeill. Selecting and reporting what is interesting. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 495 – 515. AAAI Press, Menlo Park, CA, 1996.

[27] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, New York, 1993.

[28] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 432 – 444, Zurich, Swizerland, 1995.

[29] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley Publishing Company, Reading, MA, 1977.