Data Mining as Selective Theory Extraction in Probabilistic Logic

Manfred Jaeger Heikki Mannila* Emil Weydert
MPI Informatik, Im Stadtwald
D-66123 Saarbrücken, Germany
{jaeger,mannila,weydert}@mpi-sb.mpg.de

1 Introduction

Currently, data mining is more or less a collection of different techniques and tools for various types of data sets. Imielinski [Imi95] has compared the situation in data mining to the status of database processing in the 1960's, prior to the advent of high-level query languages: then one had to write a separate application program for each query.

In this paper we take some steps towards developing a framework for representing and analyzing a large collection of data mining tasks in a principled and unified manner. One may say that the contribution that high-level query languages have made for classical database processing has two different aspects: a conceptual one, and an algorithmic one. The conceptual aspect is the insight that a wide class of relevant queries can be expressed by formal expressions constructed according to a small set of syntax rules. The algorithmic aspect is the development of universal algorithms that can efficiently process any query formulated in that syntax. At the present stage, we aspire to contribute to the theory of data mining insights that are somewhat analogous to the conceptual aspect mentioned above, not the algorithmic one. Given the huge diversity of conceivable data mining tasks (which include almost arbitrarily ambitious ones), our general representational framework can not be expected to give rise to general algorithmic methods as well.

The data mining concept that we will develop is based on arguing that

1. the task of data mining can be seen as the problem of extracting the interesting part of the logical theory of a model; and

2. the theory of a model should be formulated in a logic able to express quantitative knowledge and approximate truth.

*Permanent address: Department of Computer Science, University of Helsinki, P.O. Box 26, FIN-00014 Helsinki, Finland. Heikki.Mannila@cs.helsinki.fi. Work supported by the Academy of Finland and the Alexander von Humboldt Stiftung.

This position paper is organized as follows. In Section 2 we present the formulation of data mining as the task of finding the theory (in some logic) of a model. Section 3 argues that work done in (theoretical) AI provides a good foundation for this approach, and describes a Bacchus-Halpern type of logic $\mathcal{LDM}$ (Logic for Data Mining) that seems to be well applicable for data mining tasks in the theory formulation. Section 4 points to research topics that emerge from our data mining concept.

2 Data mining as theory extraction

Let us first consider one prototypical example of a data mining problem: the finding of *association rules* [AIS93],[AMS+96]. Suppose that $\mathbf{r}$ is a relational database of n tuples over a relation schema R such that each attribute $A \in R$ has domain $\{0,1\}$. Let X be a subset of R. Denoting $t(X) = 1$ iff tuple $t \in \mathbf{r}$ has a 1 in each attribute $A \in X$, we define the support $s(X)$ of X to be $|\{t \in \mathbf{r} \mid t(X) = 1\}|/n$. Let $B \in R$, $\sigma, \gamma \in [0,1]$. The association rule

$$X \Rightarrow B\,[\sigma, \gamma] \tag{1}$$

(with σ the *support* and γ the *confidence* of the rule) then holds in $\mathbf{r}$, iff $s(X \cup \{B\}) \geq \sigma$, and $s(X \cup \{B\})/s(X) \geq \gamma$. The data mining task of finding association rules then consists of finding expressions (1) that are true in $\mathbf{r}$. However, we do not want to list every such expression: only when support and confidence of the rule are sufficiently high, will we find the rule *interesting* and want an algorithm to include the rule in its output.

Abstracting from this specific example we are led to the following formulation of data mining problems in general: given a database $\mathbf{r}$, a language $\mathcal{L}$ for expressing statements about the data, and a criterion for distinguishing interesting statements about the data from uninteresting ones, we want to find all the interesting statements true for $\mathbf{r}$.

Formally, this means finding the set

$$ThI(\mathcal{L}, \mathbf{r}) = Th(\mathcal{L}, \mathbf{r}) \cap I(\mathcal{L}, \mathbf{r}),$$

where $Th(\mathcal{L}, \mathbf{r}) = \{\varphi \in \mathcal{L} \mid \mathbf{r} \models \varphi\}$ is the set of sentences of $\mathcal{L}$ true in $\mathbf{r}$, and $I(\mathcal{L}, \mathbf{r}) \subseteq \mathcal{L}$ is the set of interesting formulas of $\mathcal{L}$ about $\mathbf{r}$.

This *theory extraction* formulation is either explicitly or implicitly used in a variety of data mining studies [MR86, DB93, Klo95, AMS+96]. Its roots are in the use of *diagrams* of models in model theory.[1] It is based on the view that data mining is principally *descriptive*: the task is to obtain a collection of statements about the data.

In the theory extraction formulation, there are two ways of delimiting the sentences that form the description of the model. The first is the choice of the language $\mathcal{L}$, the second is the interestingness predicate $I(\mathcal{L}, \mathbf{r})$.

Interestingness for different data mining problems may be defined in a great variety of ways. We here mention a few examples to illustrate the versatility and the importance of the concept. The perhaps simplest way in which interestingness may defined is by a user-supplied restriction on the syntactic form of the statements. The association-rule example above falls into this category: a rule $X \Rightarrow B\,[\sigma, \gamma]$ is considered interesting iff $\sigma \in [p, 1]$, $\gamma \in [q, 1]$ for some $p, q \leq 1$. The user may restrict the syntax of rules considered interesting further by, for example, requesting that X contain a certain fixed subset of attributes from R.

Another way to define interestingness is through some concept of a-priori expectation: a statement is interesting if it is unexpected. As an example consider the situation where we have an up-to-date dataset $\mathbf{r}$ as well as an older version $\mathbf{s}$ of the same database from which $\mathbf{r}$ was obtained by deleting (some) and adding (many) tuples. We may expect that most general statements that were true in the old database remain true in the new version, and be particularly interested in statements for which this is not the case, and which thereby describe some qualitative change in the data. Thus, we might define: $I(\mathcal{L}, \mathbf{r}) := \mathcal{L} \setminus Th(\mathcal{L}, \mathbf{s})$.

Interestingness as unexpectedness can also be defined in a very general, completely logical way, devoid of any user-input: Fagin (cf. [Fag90]) calls a first-order sentence φ "very uninteresting" if it is true in large finite random structures with asymptotic probability 1, i.e. if we look at all possible structures over the domain $\{1, \ldots, n\}$ (there are only finitely many), and the proportion of structures in which φ holds tends to 1 as $n \to \infty$, then φ is very uninteresting. In a similar way we can derive a general criterion for interestingness in the present context: one condition for interestingness of $\varphi \in \mathcal{L}$ may often be that the probability for φ to hold in a randomly generated database tends to 0 as the size of the database tends to ∞.

In principle, one might incorporate the given concept of interestingness directly into the choice of $\mathcal{L}$, and, indeed, there is always some choice between using a richer background language $\mathcal{L}$ and a more complicated definition of interestingness on the one hand, and using a more restricted language $\mathcal{L}$ with a simpler concept of interestingness on the other. However, we feel that usually a natural distinction can be made between a more static part of the specification of a data mining task reflected in the choice of $\mathcal{L}$, and a more dynamic part reflected in the definition of interestingness.

Given our general description of data mining problems, we can identify three distinct subtasks that a data mining system has to deal with.

1. we have to be able to check the *validity* of $\varphi \in \mathcal{L}$ in $\mathbf{r}$, i.e., decide the relation $\mathbf{r} \models \varphi$;
2. we have to check whether φ is *interesting*, and
3. we must have some method of searching for candidate sentences φ that are to be checked for interestingness and validity. Particularly, this may mean to identify *plausible* sentences φ, i.e. those that by some token are likely to be true in $\mathbf{r}$.

The first two of these subtasks are mostly *deductive* in nature, whereas the third task will usually have to be performed by *inductive* methods.

One method for finding plausible φ is the use of *sampling* techniques: rather than considering the whole database $\mathbf{r}$ at once we may first look at a small random sample $\mathbf{s} \subset \mathbf{r}$ of the data. The result of a validity check for φ in $\mathbf{s}$ then can be used to evaluate the likelihood for φ to be true in $\mathbf{r}$ (cf. [KM94]). Also, the part of $Th(\mathcal{L}, \mathbf{r}) \cap I$ already computed may be used to further direct our search: clearly, we can discard φ when $\psi \in Th(\mathcal{L}, \mathbf{r}) \cap I$ with $\models \psi \to \neg\varphi$ has already been found.

Our theory extraction concept of data mining is a natural one that is directly applicable to such problems as finding association rules or *functional dependencies*. Its division into three distinct components is a conceptualization that may prove helpful for identifying and solving characteristic problems of data mining. We do not claim that in building actual data mining systems it is always advisable to use an architecture with distinct subroutines for each of these components.

3 A logic for data mining

Proceeding from the theory extraction formulation, what, then, is a good logic $\mathcal{L}$? Obviously, for each specific data mining task one chooses only a small language. Thus the above question might be more accurately formulated as: what is a good language $\mathcal{L}$ such that many data mining tasks can be usefully formulated as theory extraction in a suitably defined subclass of $\mathcal{L}$?

[1] Following Imielinski, one could also view the task of data mining as *querying* $ThI(\mathcal{L}, \mathbf{r})$.

The success of tuple relational calculus for querying relational databases, and the ability of expressing powerful integrity constraints using the same formalism, suggest that tuple relational calculus is a good starting point for such a logic.

However, there is an important ingredient missing from the tuple relational calculus. It uses a language that talks about fully specified relations between tuples. For data mining tasks, the information we want to extract from a database rarely takes the form of strict laws. More often we look for quantitative information, e.g. that a given rule applies to a certain portion of the entries in the database, or for statements that are only approximately true, e.g. that might be true in the domain from which the data is taken, but are not strictly valid in the database due to noisy and imprecise measurements. Being thus of central importance, we make statements about a qualified validity of formulas an integral part of the language.[2] It turns out that such extensions have been considered in detail in AI literature [Bac90b],[Hal90].

3.1 Syntax

For simplicity, we consider a database over a single *relation schema* $R = \{A_1, \ldots, A_n\}$ of attributes A_i. Each attribute A_i has a *domain* D_i, which is a structure on which functions and relations may be defined. A database $\mathbf{r}$ over R is a set of *tuples* $t = (t[A_i], \ldots, t[A_n])$, where $t[A_i] \in D_i$ for all i.

The tuple relational calculus, denoted by $\mathcal{TRC}$, is constructed from variables $s, t, \ldots$ ranging over the tuples in the database, constant symbols for every element of D_i $(i = 1, \ldots, n)$, symbols for the functions and relations defined on the D_i, and the attribute symbols $A_1, \ldots, A_n$.

Terms are built by the rules: every constant symbol for an element of D_i is a term of sort D_i. If $\sigma_1, \ldots, \sigma_m$ are terms (of sort D_i), and f is an m-ary function symbol for D_i, then $f(\sigma_1, \ldots, \sigma_m)$ is a term of sort D_i. If s is a variable, and A_i an attribute symbol, then $s[A_i]$ is a term of sort D_i.

Atomic formulas are either of the form $T(\sigma_1 \ldots \sigma_m)$, where σ_j is a term of sort D_i, and T is an m-ary relation symbol on D_i, or of the form $\sigma = \tau$ with σ and τ are terms of sort D_i.

By closing the language defined so far under boolean connectives ($\wedge$ and $\neg$) and quantification ($\exists$ and $\forall$) over the tuple variables one obtains the usual tuple relational calculus $\mathcal{TRC}$.

Next, we extend this language by a construct that allows us to represent statements about the approximate truth of formulas. Syntactically this additional construct is very similar to the statistical probability terms introduced by Bacchus [Bac90a]. Semantically, however, we have in mind a wider range of possible interpretations for these new terms.

An *error term* is either a constant symbol q for some $q \in \mathbb{Q}$ (or, when necessary, a countable extension of $\mathbb{Q}$), or an expression of the form $G(\chi(\boldsymbol{s}) \mid \psi(\boldsymbol{s}))$, where $\chi(\boldsymbol{s}), \psi(\boldsymbol{s})$ are formulas in $\mathcal{TRC}$ (i.e., not containing any error terms themselves!) whose free variables are among $\boldsymbol{s} = (s_1, \ldots, s_k)$.

An atomic error formula is any expression of the form $\eta \leq \delta$ or $\eta = \delta$, with error terms η and δ.

Our final representation language $\mathcal{LDM}$ is obtained by again closing the union of $\mathcal{TRC}$ and the set of atomic error formulas under boolean connectives and quantification.

For an example of an $\mathcal{LDM}$ expression, let R, X, B, σ, γ be as in the association rule example from section 2. Then

$$G(t[B] = 1 \mid \bigwedge_{A \in X} (t[A] = 1)) \geq \gamma \qquad (2)$$

is an $\mathcal{LDM}$-expression with the intended meaning that the association rule $X \Rightarrow B$ has confidence γ. Similarly, the formula

$$G(\bigwedge_{A \in X} (t[A] = 1) \wedge (t[B] = 1)) \geq \sigma \qquad (3)$$

is meant to represent the fact that the rule $X \Rightarrow B$ has significance σ. Here we have used a convention to simply write $G(\chi(\boldsymbol{s}))$ instead of $G(\chi(\boldsymbol{s}) \mid \tau(\boldsymbol{s}))$ where $\tau(\boldsymbol{s})$ is a tautology.

In the following section we provide $\mathcal{LDM}$ with a semantics in which (2) and (3) will have the intended meaning.

3.2 Semantics: error measures

The definition of the relation $\mathbf{r} \models \varphi$ is straightforward for any $\varphi \in \mathcal{TRC}$. To also define this relation for arbitrary $\varphi \in \mathcal{LDM}$ we have to explain the meaning of an error term $G(\chi \mid \psi)$.

A somewhat vague, yet useful, guiding intuition is that such a term represents the "degree of falsity" of χ given that ψ is true. Formally, we use error measures to capture this intuition.

Let R be a relational schema. An *error measure* for R is any function g that assigns to every triple $(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s}))$, where $\mathbf{r}$ is a database over R, and $\chi(\boldsymbol{s}), \psi(\boldsymbol{s})$ are formulas of $\mathcal{TRC}$, a value $q \in \mathbb{R} \cap [0, 1]$ such that

1. $g(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s})) = 0$ if $\mathbf{r} \models \psi(\boldsymbol{s}) \rightarrow \chi(\boldsymbol{s})$, and $\mathbf{r} \models \exists \boldsymbol{s} \psi(\boldsymbol{s})$.
$g(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s})) = 1$ if $\mathbf{r} \models \neg \exists \boldsymbol{s} \psi(\boldsymbol{s})$.

2. $g(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s})) \leq g(\mathbf{r}, \chi'(\boldsymbol{s}), \psi'(\boldsymbol{s}))$, if $\models \psi(\boldsymbol{s}) \leftrightarrow \psi'(\boldsymbol{s})$ and $\models \chi'(\boldsymbol{s}) \rightarrow \chi(\boldsymbol{s})$

[2] Another alternative would be using tuple relational calculus augmented with aggregate functions; the foundations of such languages, however, do not seem as simple as in the other alternative.

Given an error measure g, the interpretation of an error term $G(\chi(\boldsymbol{s}) \mid \psi(\boldsymbol{s}))$ in $\mathbf{r}$ is the value $g(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s}))$, and the relation $\mathbf{r} \models \varphi$ (more precisely: $\mathbf{r} \models_g \varphi$) is defined in the obvious manner for $\varphi \in \mathcal{LDM}$.

The definition for error measures given here is about as general as can be. The conditions 1. and 2. listed here are minimal requirements for how an error measure behaves with respect to logical properties of its arguments. These conditions will also be met by many functions that we would not consider as sensible error measures. We shall leave it to some later work to describe further reasonable conditions for error measures.

We now turn to different kinds of error measures, and how they may be utilized in the formalization within $\mathcal{LDM}$ of interesting data mining problems.

Some error measures can be naturally defined in terms of the subsets of tuples in $\mathbf{r}$ defined by χ and ψ, i.e. the sets $\mathbf{r}(\chi(\boldsymbol{s})) := \{(t_1, \ldots, t_k) \mid t_i \in \mathbf{r},\ \mathbf{r} \models \chi[t_1, \ldots, t_k]\}$ and $\mathbf{r}(\psi(\boldsymbol{s})) := \{(t_1, \ldots, t_k) \mid t_i \in \mathbf{r},\ \mathbf{r} \models \psi[t_1, \ldots, t_k]\}$.

One such error measure is g_1, defined as

$$g_1(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s})) := \frac{\mid \mathbf{r}(\psi(\boldsymbol{s}) \wedge \neg\chi(\boldsymbol{s})) \mid}{\mid \mathbf{r}(\psi(\boldsymbol{s})) \mid}$$

if $\mid \mathbf{r}(\psi(\boldsymbol{s})) \mid > 0$ and $g_1(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s})) := 1$ else.

Observe that in the special case that $\boldsymbol{s} = \emptyset$ we get $g_1(\mathbf{r}, \chi, \psi) = 0$ if $\mathbf{r} \models \psi \wedge \chi$, and $g_1(\mathbf{r}, \chi, \psi) = 1$ otherwise. Compare this to the behaviour of the term $g_1(\mathbf{r}, \chi \to \psi)$ (i.e. $g_1(\mathbf{r}, \chi \to \psi, \tau)$ with τ a tautology), which equals 0 if $\mathbf{r} \models \psi \to \chi$ and 1 if $\mathbf{r} \models \psi \wedge \neg\chi$.

g_1 is the appropriate error measure for giving the intended semantics to (2) and (3):

$$\mathbf{r} \models_{g_1} (2) \wedge (3)$$

iff the association rule $X \Rightarrow B[\sigma, \gamma]$ holds in $\mathbf{r}$.

Another natural error measure is g_3, which, roughly speaking, is inversely proportional to the size of the maximal sub-database $\mathbf{r}' \subseteq \mathbf{r}$ in which the implication $\forall \boldsymbol{s}(\psi(\boldsymbol{s}) \to \chi(\boldsymbol{s}))$ is true. Precisely, for given $\mathbf{r}, \psi, \chi$ let

$$m := max\{\mid \mathbf{r}' \mid \mid \mathbf{r}' \subseteq \mathbf{r},\ \mathbf{r}' \models \forall \boldsymbol{s}(\psi(\boldsymbol{s}) \to \chi(\boldsymbol{s}))\},$$

and put

$$g_3(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s})) := 1 - \frac{m}{\mid \mathbf{r} \mid}.$$

Observe that here $g_3(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s})) = g_3(\mathbf{r}, \chi(\boldsymbol{s}) \to \psi(\boldsymbol{s}))$.

g_3 is one way to let an error measure represent the minimal amount of change necessary to transform the database $\mathbf{r}$ into a database $\mathbf{r}'$ with $\mathbf{r}' \models \psi \to \chi$ – here simply by deleting tuples. This, however, is not the only kind of manipulations that one may consider. Alternatively, we may change some attribute values of some tuples in the database. In case that we are given some metric d_i on every domain D_i of attributes A_i that we are allowed to change, we can then quantify in numerous ways the amount of change necessary to transform $\mathbf{r}$ into $\mathbf{r}'$.

Suppose, for instance, that $\mathbf{r}'$ is obtained from $\mathbf{r}$ by changing the attribute values of $A_1, \ldots, A_k$ in some or all tuples of $\mathbf{r}$, and that d_i is a metric on D_i for $i = 1, \ldots, k$. Then we may let

$$c(\mathbf{r}, \mathbf{r}') := max\{d_i(t[A_i], t'[A_i]) \mid t \in \mathbf{r},\ i = 1, \ldots, k\}, \quad (4)$$

where t' is the transformation of t. Putting $c(\mathbf{r}, \mathbf{r}') := 1$ if $\mathbf{r}'$ can not be obtained from $\mathbf{r}$ by such changes, we then let

$$g_d(\mathbf{r}, \chi(\boldsymbol{s}), \psi(\boldsymbol{s})) := inf\{c(\mathbf{r}, \mathbf{r}') \mid \mathbf{r}' \models \forall \boldsymbol{s}(\psi(\boldsymbol{s}) \to \chi(\boldsymbol{s}))\}. \quad (5)$$

Note that this error measure only is defined for a specific underlying relation schema, whereas g_1 and g_3 may be applied to any relation schema.

As an example for how error measures of the type (5) may be applied, we consider *approximate functional dependencies*.

Let R be a relation schema, $X, Y \subseteq R$. A functional dependency $X \to Y$ holds in an R-database $\mathbf{r}$ if for all $s, t \in \mathbf{r}$: $s[A_i] = t[A_i]$ for all $A_i \in X$ implies $s[A_j] = t[A_j]$ for all $A_j \in Y$. Now suppose that every domain D_j with $A_j \in Y$ is equipped with a metric (for simplicity, assume, for instance, that $D_j = \mathbb{R}$ for all j). Then we may speak of an "approximate functional dependency with accuracy ϵ" when $s[A_i] = t[A_i]$ for all $A_i \in X$ implies $d_j(s[A_j], t[A_j]) \leq \epsilon$ for all $A_j \in Y$.

We can now represent the statement that $X \to Y$ is a functional dependency with accuracy ϵ by the $\mathcal{LDM}$-expression

$$G(\bigwedge_{A_i \in X} (s[A_i] = t[A_i]) \to \bigwedge_{A_j \in Y} (s[A_j] = t[A_j])) \leq \epsilon,$$

which has the intended meaning when G is interpreted by the error measure g_d obtained from the d_j $(A_j \in Y)$ through (4) and (5).

While the representation language $\mathcal{LDM}$ here described already is quite powerful, there are a multitude of possibilities to further increase its expressiveness.

To mention a few: we might introduce functions and relations on the tuples themselves. This is not a real extension, though: such relations and functions can always be encoded in an additional attribute. A more fundamental extension would consist in the introduction of variables ranging over the domains and quantification over these variables. We have chosen not to include domain variables in our language because this would make $\mathcal{LDM}$ undecidable whenever the underlying relation schema R contains attributes with a domain

whose first-order theory is undecidable (as, for instance, the natural numbers with addition and multiplication). As it is defined here, $\mathcal{LDM}$ is decidable whenever the interpretation of the error terms $G(\cdot,\cdot)$ is computable.

Some further extensions have not been introduced here mostly for simplifying the exposition, but may be considered in the future. These include selective quantification of the variables in the error terms, i.e. using terms $G\boldsymbol{s}'(\chi(\boldsymbol{s}),\psi(\boldsymbol{s}))$ for some $\boldsymbol{s}' \subseteq \boldsymbol{s}$ in which only the variables $\boldsymbol{s}'$ are bound, while those in $\boldsymbol{s} \setminus \boldsymbol{s}'$ remain free, as well as the combination of error terms of different kinds $G_1(\cdot,\cdot), G_2(\cdot,\cdot), \ldots$ to be interpreted by different error measures, and nested error terms.

4 Research questions

The general framework of $\mathcal{LDM}$ gives rise to the formulation and investigation of theoretical questions about characteristic data mining problems.

- What is the complexity of querying $ThI(\mathcal{L},\mathbf{r})$, as a function of the logical complexity of $\mathcal{L}$. (Note: deciding whether a formula φ of $\mathcal{LDM}$ is true in database $\mathbf{r}$ can be solved in time polynomial in the size of the database – provided that evaluation of error-terms $G(\chi \mid \psi)$ is polynomial.)
- What types of quantitative information do we seek to extract in data mining; and can it always be encoded by error measures? What are the most useful error measures, and what are their common properties?
- Which inductive methods can be used search efficiently for sentences in $ThI(\mathcal{L},\mathbf{r})$? See also [MT96] for some simple general complexity results in this context.
- Is it possible to develop a theory of 'data mining task optimization', which would somehow resemble the theory of query optimization?

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *sigmod93*, pages 207 – 216, May 1993.

[AMS+96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, Menlo Park, CA, 1996.

[Bac90a] F. Bacchus. Lp, a logic for representing and reasoning with statistical knowledge. *Computational Intelligence*, 6:209–231, 1990.

[Bac90b] F. Bacchus. *Representing and Reasoning With Probabilistic Knowledge*. MIT Press, 1990.

[DB93] Luc De Raedt and Maurice 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.

[Fag90] R. Fagin. Finite model theory – a personal perspective. In S. Abiteboul and P. Kanellakis, editors, *Proceedings 1990 International Conference on Database Theory*, number 470 in Springer Lecture Notes in Computer Science, 1990.

[Hal90] J.Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

[Imi95] Tomasz Imielinski. Invited talk at KDD-95. 1995.

[Klo95] Willi Kloesgen. Efficient discovery of interesting statements in databases. *Journal of Intelligent Information Systems*, 4(1):53 – 69, 1995.

[KM94] J. Kivinen and H. Mannila. The power of sampling in knowledge discovery. In *Proceedings of the 13th ACM Symposium on Principles of Database Systems (PODS 94)*, 1994.

[MR86] Heikki Mannila and Kari-Jouko Räihä. Design by example: An application of Armstrong relations. *Journal of Computer and System Sciences*, 33(2):126 – 141, 1986.

[MT96] Heikki Mannila and Hannu Toivonen. On an algorithm for finding all interesting sentences. In *Proc. ECMSCR'96*, 1996. To appear.