Bayesian Networks in a Data Mining tool

Robert Castelo   Arno Siebes

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
robert@cwi.nl   arno@cwi.nl

Abstract

The growing area of Data Mining defines a general framework for the induction of models from databases. Bayesian Networks are a class of graphical models which are able to deal with uncertainty. Nowadays, they are among the most promising ones. This paper summarizes our work on recovering Bayesian Networks in the framework of Data Mining, by commenting and discussing the insight gained in developing a Bayesian Network induction module in an existing Data Mining tool, Data Surveyor.

1 Introduction

The way we represent *knowledge* in a certain domain, is of fundamental importance in the ability of dealing automatically with that knowledge in any knowledge based system. Graphical models have become a very promising way of representing domain knowledge, and efforts, around this models, have been carried out during the last ten years [25, 18].

Every type of model uses a different way of understanding data, and many applications have been developed to induce these models, and shown their merits. But the management of these models from a common point of view, is an improvement of their benefits. Quite often we are completely uninformed about our data, and we need to induce different types of models to get a clear picture of the biases contained in it. This situation lead us to the idea that we need to induce and compare easily different class of models, by using a common inductive query language. Bayesian Networks have already been applied in some fields of science [11], but we think that they only can show their full abilities, within this common point of view, Data Mining.

This paper summarizes the work carried out in [6], in the following way. First we will define briefly, a Bayesian Network. Then, we will describe the architecture of a Data Mining tool where we have developed our ideas. We will outline how do we fit Bayesian Networks within this architecture, and we will show some small, but interesting, applications of inducing these models using this new developed part. Finally, we will point out some conclusions and further lines of work.

2 Bayesian Networks

A Bayesian Network is a probabilistic model of a certain domain described as a set of variables and their relationships. More precisely, networks are models that give us a simple interpretation for some part of a concrete domain, and they combine well with the bayesian interpretation of probability.

In Bayesian Networks, the graphical representation is given by Direct Acyclic Graphs (DAGs), where a node is a variable, and each edge states a relationship of conditional dependence between two nodes. In other words, the lack of an edge, states a relationship of (conditional) independence.

A given domain $U = \{x_1, \ldots, x_n\}$ may be described in terms of a joint probability distribution

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \ldots, x_{i-1})$$

and the set of assertions of (conditional) independence, given by the Bayesian Network, reduces the previous expression $P(x_i|x_1, \ldots, x_{i-1})$ to $P(x_i|\pi_i)$ for some subset of variables $\pi_i \subseteq U$. This set π_i

is the *parent set* of the variable x_i in the DAG. In general, this parent set π_i will be a proper subset of $\{x_1, \ldots, x_n\}$. Formally, a Bayesian Network B is the pair

$$B = (B_S, B_P)$$

where B_S is the network structure and B_P is the set of parameters or (conditional) probability distributions associated to every variable.

This last definition introduces two of the tasks we may perform in learning Bayesian Networks. One is to recover the structure B_S, and the other is to recover the (conditional) probability distributions B_P. The most interesting task, is to recover the structure B_S, because this amounts to the induction of a dependency/graphical model.

Moreover, the task of recovering a Bayesian Network structure is the most complex one, due to the number of possible DAGs to be recovered, which grows exponentially, in the number of nodes.

2.1 Recovering the structure

We identify two main approaches for recovering the structure of a Bayesian Network. One makes assumptions on the topology underlying the dataset. These assumptions yield algorithms of polinomic complexity. The second approach does not put any constraint on the structure to be recovered. It uses search trategies, which make feasible to explore the vast search space. We may find proposals in the first approach in [12, 5, 8, 19, 20].

These two approaches share the goal of trying to recover a dependency model, but while the graphical assumptions of the first one, restrict the possibility of recover the true structure in most of cases, the second one always works towards the achievement of the true structure.

The main merit of the first approach is that overcomes the complexity of the problem. And some authors [21] assert that performing inference over this more *simple* structure, the values obtained, are similar to those we would get inferring over the true structure.

Anyway, we are more interested in trying to recover the structure closest to the true one.In this second approach, every algorithm sets up a criterion of *goodness* of a Bayesian Network. This criterion is commonly called, a *quality measure*. This measure guides the search process, which is scheduled by some search strategy.

These algorithms are usually divided according to the type of quality measure they use. We point out three approaches: bayesian measure based algorithms [10, 3], information criterion based algorithms [9, 2], and minimum description length principle based algorithms [15]. A comparison between these algorithms is beyond the scope of this paper, but the reader may consult [2, 22, 21] for a discussion of their properties.

Our point is that any of these model induction algorithms can be cast in terms of search strategy. This, in turn, can be further split into three components: heuristic function that evaluates a tentative solution (partial model) by using a goodness measure; a set of operators for creating new solutions; and a search strategy (hill-climbing, beam-search, etc). By combining this aspects (evaluation function, search operators, and search regime) a wide variety of algorithms can be described.

3 A Data Mining tool: Data Surveyor

The previous formalization of model induction algorithms is the base of the architecture of the Data Mining tool Data Surveyor. This tool is currently developed as part of the ESPRIT-IV project KESO (Knowledge Extraction for Statistical Offices; 1996-1999). It has been our target architecture, where we have developed a new module, which makes the tool able to recover Bayesian Network structures.

Data Surveyor is structured in three layers: the user interface (the frontend), the discovery layer (the mining server) and the data warehouse (the backend). These layers can work on separate machines connected through a TCP/IP network. The data warehouse may be built and maintained by some of the existing commercial platforms in the market, but by default Data Surveyor incorporates the Monet Database Management System. Monet [1] is an extensible parallel database kernel that has been developed at the University of Amsterdam and CWI since 1993. Its design is based on trends in hardware technology: main memories of hundreds of megabytes are now affordable, and custom CPUs can perform over 200 MIPS. Monet has already achieved considerable successes in Data Mining [16].

We may see a sketch of the architecture of Data Surveyor in figure 1.

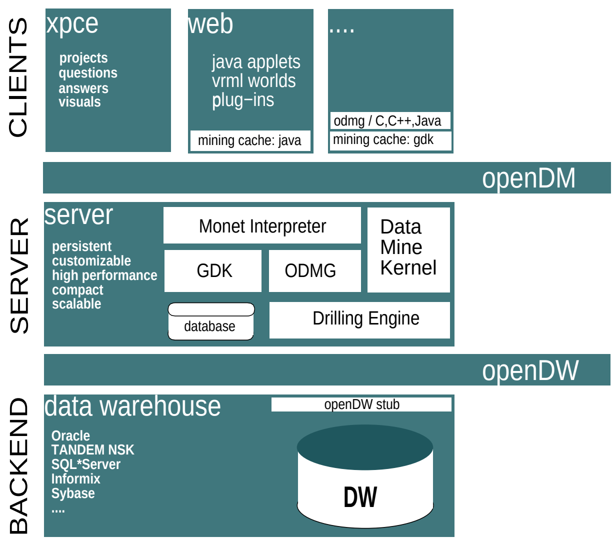


Figure 1: Architecture of Data Surveyor

The mining server supports access from several clients at the same time. The interaction between layers is possible through two different protocols (openDM for client-server, and openDW for server-data warehouse). Two of the main merits of this architecture are: the ability to mine large databases, by using a combination of hardware/DBMS prepared to deal with them, and the persistent storage of the search space, by using the DBMS for this purpose as well.

By persistent storage of the search space, we mean to store the tentative solutions (we call them below *hypotheses*) that are being generated to explore the search space. In other words, the persistent storage of the search space refers to the persistent storage of the *explored* search space. Even being smaller than the whole search space, this *explored* search space might be really huge, but every day the magnetic memories become bigger, faster and cheaper. And this makes possible to store and access efficiently any point of the *explored* search space.

The ability of storing the search space is very important if we want to mine large databases.In that case, the mining processes may take long time, and the user might desire to stop the process momentarily. Then this user may assess the current state of the search, and eventually decide to continue this search from some other point changing some parameters.

The most important parts to be developed in order to let Data Surveyor construct Bayesian Networks models are contained in the Mining Server. In figure 2 we may see the internal structure of the Mining Server.

The models to be induced, are described within this architecture through a *description language*, which is implemented using the ODMG standard. A concrete *description language* binds a set of operators and a set of quality measures. The main elements of the Mining Server are:

- The *Search Manager*, which contains several search modules that implement different search strategies. Within the architecture of Data Surveyor, it is established the naming convention in which we will identify *hypothesis* with model, more concretely, in our case a DAG. Every search module uses a certain operator to go through the search space.

- The *Description Generator*, implements the operators. Some of them create initial hypotheses, and the rest derive new hypotheses by manipulation of already explored hypotheses.

- The *Quality Computer*, implements a wide variety of *goodness* measures. Each one is associated to a concrete description language, and computes the qualities of hypotheses using aggregated information from the backend database.

- The *Mining Conductor*, which manages and schedules mining tasks, and implements routines to create new mining projects, search spaces, search tasks, etc.

- The *Search Space Manager*, is the storage manager of the Mining Server for the search space. It can maintain different hypotheses spaces for several tasks at the same time.

Below we may see the algorithmics of a mining task. The condition of existence of an open path, refers to the possibility of following one or more

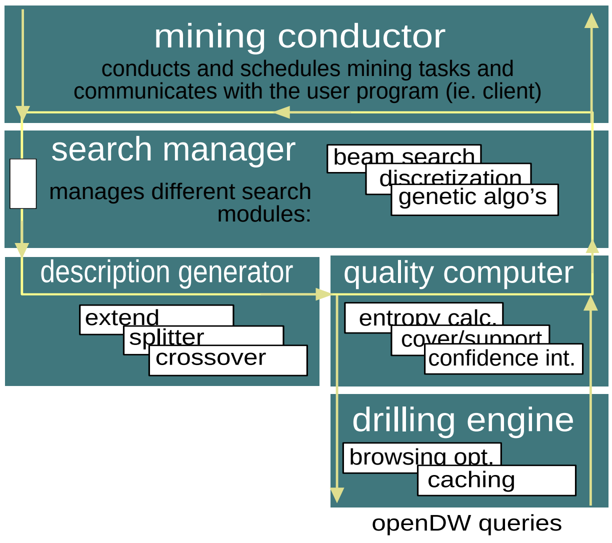


Figure 2: Flow of the hypotheses within the Mining Server

branches in the search, that improve the quality of the current best hypothesis, according to the current search strategy. When the Quality Computer ends the qualification task, the Search Manager can continue its job, because new qualified hypotheses have arrived to the search space.

```
procedure mining_task is
  While ∃ open_path do

      Search Manager selects a set of hypotheses
      from the search space to be improved, according
      to the search strategy selected. For each
      hypothesis, picks a hypothesis generating operator,
      which will create a new set of candidates to explore.

      Description Generator executes the chosen
      operator on the given set of hypotheses and
      generates new hypotheses.

      Quality Computer takes newly generated
      hypotheses and computes their qualities using
      the drilling engine to interact with the
      database server.

  endwhile
endprocedure
```

Other important aspect, is the concept of *neighbourhood*, which is introduced by the Description Generator. This component, works on a set of parent hypotheses, and use them to produce another set of *neighbour* hypotheses. In other words, those models that are similar, and might lead to a better one.

4 Updating Data Surveyor

Mainly, the two components that should be updated for recovering Bayesian Networks are the Description Generator, and the Quality Computer. Within the Description Generator we have developed an INIT operator, which creates an initial hypothesis, which may be either an empty DAG (a DAG with no arcs), or an arbitrary DAG contained in the search space and picked up by the user. Any new mining task starts from this initial hypothesis. Moreover, a NEIGHBOUR operator has had to be developed. This NEIGHBOUR operator, creates a neighbourhood from a given hypothesis. This neighbourhood is formed by all possible DAGs with one arc more, one arc less, and one arc reversed, and it is stored in the search space. To keep the new generated graphs *acyclic*, the operator tests whether the addition or reversion of an arc can introduce a cycle.

The shape of this neighbourhood makes possible to take any direction in the search space. This is an important feature, because it helps to the search strategies to escape many local maxima. On the other hand, the generation of this neighbourhood is actually the bottleneck of the system. Which does not mean that the system cannot work properly, but we have assessed that to double the amount of tuples of a dataset does not mean to double the recovering time. For databases with large schemas the recovering time, mainly stems from the application of the NEIGHBOUR operator.

Concerning the Quality Computer, we have implemented the Bayesian Measure of Cooper and Herskovits [10], which basically works by computing *count* aggregates on data cubes [23]. Every data cube is formed by a parent set of a given variable plus this variable.

These count aggregates refer to a given variable v_i and its parent set π_i, in the following form:

- N_{ijk} number of cases where v_i takes the domain value x_{ik} and π_i takes the configuration of values $x_{\pi_i j}$

- $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ where r_i is the number of possible values that v_i can take

The quality for a given variable is computed upon a formula which works using these count aggregates, and the quality of a whole DAG is computed by adding up to the results of every variable. So, the Bayesian Measure (and most types of measures) works as a local quality for every variable.

To take profit of this feature the Quality Computer splits up every DAG in subDAGs. Each subDAG has a unique sink, and it is stored as a new hypothesis. The mechanism of creating and storing hypothesis in the search space, checks out automatically whether a hypothesis already exists, to avoid to duplicate hypotheses and therefore, to introduce loops in the search process.

Further, every hypothesis has associated a quality which is stored with it. Thus, the fact of storing subDAGs makes cheaper the re-calculation of the quality of any complete DAG.

5 Experiments

In the experimentation, we have been using a search module that implements a beam search with three parameters: the maximum number of best results to be retrieved by the user, the width of the beam, and the maximum number of calls to the neighbour operator (called *depth*). These calls to the neighbour operator are performed until this maximum number is reached or the branch is rejected. By setting the first two parameters to one, and the third with no limit, we are, in fact, scheduling a Hill-Climber. By setting no limits in any of them we are scheduling an exhaustive search.

We have checked out that the ability of tunning the search by using these three parameters and the shape of the neighbourhood we have implemented, allow the search to escape from many local maxima.

5.1 Hidden variables

The following experiments have been developed towards the assessment of the existing theories of discovering hidden influences in data [13, 25], which might explain some strange patterns reflected in our data.

The experiments also, make use of some properties of conditional independence to detect the presence of a hidden variable. Thus, has been proved useful in classification domains.

We generate a dataset with three variables, with the aim of reflecting the following three clusters of tuples:

a_1	a_2	a_3
true	true	xxx
xxx	false	true
false	false	false

Where the value xxx represents either true or false. This dataset is generated upon the Bayesian Network of the figure 3, which sets up all three variables mutually dependent. The bayesian measure implemented in Data Surveyor recovers perfectly, from the dataset, the structure we put beforehand.

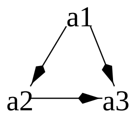


Figure 3: Bayesian Network for the clustering example

Then we started a clustering process using the the program Autoclass [7], which finds, indeed, the three previous clusters. Autoclass generates a report with the most probable class assigned to every case. We extract this information as a column which we paste to the original dataset. Let's call this new column of the dataset, C (the *class* column).

Starting again the mining process, but using these four attributes, Data Surveyor recovers a structure where the class variable renders the other three attributes conditionally independent. In other words, we may see that renders a common influence in a_1, a_2, a_3.

Further, this Bayesian Network structure is suitable to perform Bayesian Classification [17, 4]. To get a better picture of this result, we have calculated the Kullback-Leibler cross entropy [14] of the network of figure 4, which is a measure of closeness between the original distribution and the aproximated distribution given by the Bayesian Network.

In this figure it is written the value of this

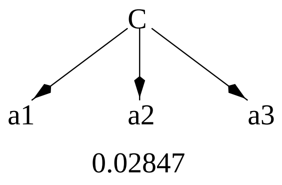


Figure 4: Bayesian Network with a hidden common influence, as a class variable. The value corresponds to the Kullback-Leibler cross entropy

Kullback-Leibler cross entropy and we may see that is close to the zero value. If we calculate this entropy related to the simplest aproximation given by assuming a_1, a_2, a_3 independent (a Bayesian Network with no arcs), we will obtain 0.27238, which is almost ten times greater than introducing this class variable. Thus, we can assert that for some datasets we can aproximate better its probability distribution by introducing some new variable which renders a common influence on some subset of the variables. And this turns in better accuracy when we will perform inference.

Instead of inferring the class column through a clustering process, let's create a dataset with this class column and the other a_1, a_2, a_3 with the topology shown in figure 4. Again, Autoclass finds three clusters of tuples, but if we start the mining process using only a_1, a_2, a_3 as sources, we will obtain the results of figure 5.

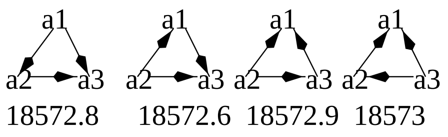


Figure 5: Recovered Bayesian Networks in the clustering example, with the corresponding values of the Bayesian Measure

In this figure the numbers below the DAGs correspond to the values of the Bayesian Measure, i.e. the quality. This quality can only be compared between networks recovered from the same dataset. It is in its logarithmic form without the minus sign. The smaller quality, the better network.

Therefore, it has been shown that, when recovering a dependency model which yields every variable mutually dependent, we may think that there exists a hidden influence which renders them *conditionally* independent [13], which, in some cases, could be used as a class variable.

We might find the situation in which a clustering process over a database with a large schema, does not produce meaningful results, because of the intervention of some *noisy* variables. If we recover a dependency model and we perform this clustering process over those set of variables that appear mutually dependent, probably we will find more meaningful clusters in the dataset formed by that attributes.

Another interesting result concerns the ability of the Bayesian Measure to recover I-Maps [25, 18] (which may be *minimal* for large databases). With this property, we can trust faithfully every recovered assertion of (conditional) independence. We do know that any formalization of conditional independence is closed under *Semi-Graphoid* axioms [18, 24]. So, it is possible to extract new, and true, assertions of conditional independence, by applying this axioms on the recovered dependency model. This new assertions may notice us of the lack of expressiveness of a current DAG, which in turn might lead to the discovery of a hidden influence.

5.2 Noise

An important problem we often find with real-world data, is the noise. There is several types of noise but we will only refer to that produced by those tuples that, for some reason, do not reflect what it has been measured. Thus, we are interested in knowing how does the Bayesian Measure behaves in front of this noise. To simulate and experiment this, we will introduce *gaussian* noise using a normal distribution, which power is ruled by the variance of this normal distribution. There is many ways of introducing this noise, and the way we have used is to disturb the selection of the corresponding tuple in the probability distribution of the dataset. In order to shift the selection of this tuple one or more entries, when the noise appears.

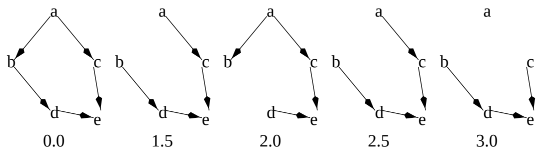
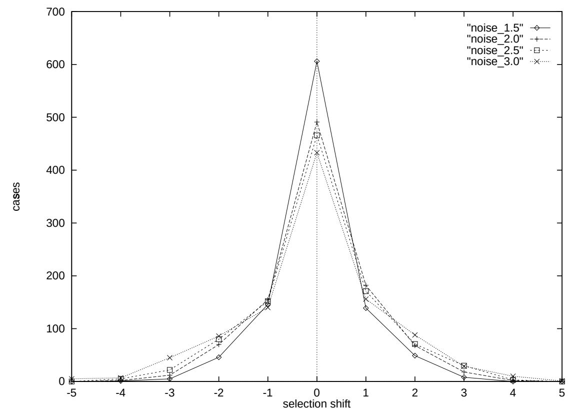


Figure 6: Bayesian Networks recovered from datasets with different levels of noise

In figure 6 we show, on the left hand, the original network, which has been recovered correctly because no noise was introduced. The other networks correspond to those recovered with different levels of noise. We should point out that we experimented with values on the variance from 0.1 to 3.0, but only from 1.5 the effects of the noise affected the recovered structure.



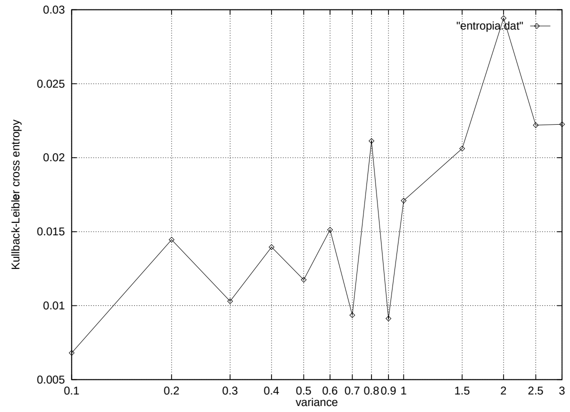


Figure 7: Noise distribution and entropy

In figure 7 we may see two plots. The first one corresponds to the normal distributions from which we introduce the noise. The x axis indicate the selection shift. And the second one corresponds to the Kullback-Leibler cross entropy computed given the original Bayesian Network and different datasets with different levels of noise. This last plot shows that as long as the noise increases, the distance between the distribution of the dataset and the distribution of the Bayesian Network, increases as well.

The domain of the five variables we have used in this experiment is bivalued, so we have a probability distribution over a possible set of tuples with 32 entries. In figure 8 we may see how the probability distribution is modified by the effect of the noise.

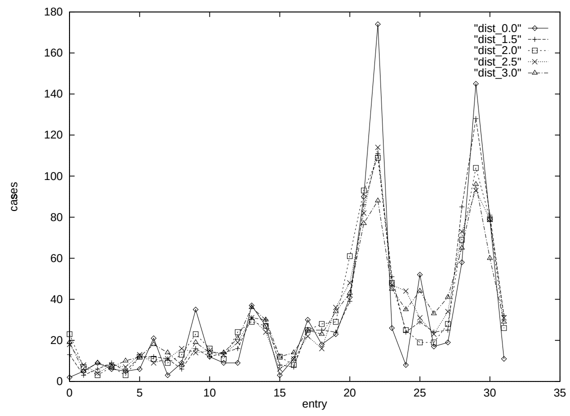


Figure 8: Probability distributions affected by the effect of noise

The way in which this noise affects the process of recovering a Bayesian Network structure, is by removing some arcs. In other words, introducing assertions of marginal independence. This means that for minimal I-maps, as the case of the figure 6, this effect destroys its I-mapness, according to the original probability distribution.

Finally, we should point out that the size of the dataset used to perform these experiments was one thousand tuples. If we look carefully the first plot of figure 7, we will realize that the amount of tuples that were affected by the noise (this means at least one variable was modified) reaches from 40% for 1.5 of variance, up to 56% for 3.0 of variance.

And the values of the noise for which the structure did not suffer any change (0.1 to 1.0), modified up to 32% of the dataset.

6 Conclusions

A Data Mining tool, as Data Surveyor, is an ideal workbench for the task of recovering dependency models for several reasons:

- It is prepared to deal with large databases
- It is able to stop the discovery process and resume it from an arbitrary point in the search

space

- In the implementation we had not to bother about how do we access the data
- It makes easier the experimentation by allowing to select an arbitrary set of source attributes and the ability to select different search strategies makes possible to schedule different recovering algorithms.

The neighbourhood operator we designed plus the parameterized beam search has let us to escape local maxima in many of our experiments. We think that this framework, will make easier to find new ways of improving the task of recovering Bayesian Networks structures, because we tackle the problem in terms of adding new search strategies, new quality measures and/or new operators that might afford different ways of exploring the search space.

The possibility of identifying patterns of hidden influences in data, is of high interest in data analysis, and any kind of automatic support to its identification can become an important feature to any Data Mining and OLAP system.

With low levels of noise, in the way we introduced it, the Bayesian Measure has a good behavior.

7 Further Research

New search approaches should be developed, towards the achievement of better local maxima. Research in quality measures also is of fundamental importance for recovering structures. It is important to try to find cheaper ways of formalizing these measures.

Research in operators that might explore the search space in different ways is important. Mainly to overcome the problem of generating the current neighbourhood of DAGs.

Algorithms that can help identifying patterns of hidden influences in data, should be developed and integrated into Data Surveyor. Its implementation form might arise from the interaction with other models (e.g. clustering), and from the explotation of the axiomatic properties of the dependency models.

The noise is an element that appear quite often in real-world data. It is important to find ways of detecting and overcome it, when we perform Bayesian Networks recovering.

Besides probability, the incorporation of other formalizations of uncertainty, e.g. possibility theory, might make the tool useful in broader areas of research in uncertainty [22].

References

[1] Peter A. Boncz and Martin L. Kersten. Monet: an impressionist sketch of and advanced database system. In *Proceedings BITWIT'95*, 1995.

[2] Remco R. Bouckaert. *Bayesian Belief Networks: from Construction to Inference.* PhD thesis, Faculteit Wiskunde en Informatica, Utrecht University, June 1995.

[3] W.L. Buntine. Theory refinement on bayesian networks. In P. Smets B.D. D'Ambrosio and P.P. Bonissone, editors, *Proceedings of Uncertainty in Artificial Intelligence*, volume 7, pages 52–60, Los Angeles, 1991. Morgan Kaufmann.

[4] W.L. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, April 1996.

[5] L.M. Campos and J.F. Huete. Efficient algorithms for learning simple belief networks. Technical report, DECSAI, Universidad de Granada, Departamento de Ciencias de la Computacion e Inteligencia Artificial, 1992.

[6] Robert Castelo. Bayesian networks in data surveyor. Master's thesis, Facultat d'Informatica de Barcelona, Universitat Politecnica de Catalunya, 1997.

[7] Peter Cheeseman and John Stutz. Bayesian classification (autoclass): Theory and results. In P. Smyth U. Fayyad, G. Piatesky-Shapiro and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 61–83. AAAI Press, Menlo Park, CA, 1995.

[8] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with depen-

dence trees. *IEEE Transactions on information theory*, 14(3):462–467, May 1968.

[9] G. Cooper and E.H. Herskovits. Kutató: an entropy-driven system for the construction of probabilistic expert systems from data. In *Proceedings of the 6th. Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1990.

[10] Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[11] A. Mamdani D. Heckerman and M. Wellman. Real-world applications of bayesian networks: Introduction. *Communications of the ACM*, 38(3):26, March 1995.

[12] Azaria Paz Dan Geiger and Judea Pearl. Learning simple causal structures. *International Journal of Intelligent Systems*, 8:231–247, 1993.

[13] D. Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, Microsoft Corporation, March 1995.

[14] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals Mathematical Statistics*, 22:79–86, 1951.

[15] Wai Lam and Fahiem Bacchus. Learning bayesian belief networks: an approach based on the mdl principle. *Computational Intelligence*, 10(3):270–293, 1994.

[16] M.L. Kersten M. Holsheimer and M.L. Mannila. A perspective on databases and data mining. In *Proceedings 1st Conference in Knowledge Discovery in Databases - KDD'95*, 1995.

[17] R. Hanson P. Cheeseman and J. Stutz. Bayesian classification theory. Technical Report FIA-90-12-7-01, Artificial Intelligence Research Branch, NASA Ames Research Center, 1990.

[18] J. Pearl. *Probabilistic Reasoning in intelligent systems*. Morgan Kaufmann, 1988.

[19] J. Cabos R. Sangüesa and U. Cortes. Possibilistic conditional independence: a similarity-based measure and its application to causal network learning. Technical report, Department of Software (LSI), Technical University of Catalonia (UPC), 1996.

[20] G. Rebane and J. Pearl. Recovey of causal poly-trees from statistical data. In T.S. Levitt L.N. Kanal and J.F. Lemmer, editors, *Proceedings 3rd. Worshop in Uncertainty in Artificial Intelligence*, volume 8, pages 175–182. North-Holland, Elsevier Science, 1989.

[21] R. Sangüesa and Ulises Cortés. Learning causal networks from data: a survey and a new algorithm for recovering possibilistic causal networks. *AI Communications*, 10(1):31–61, March 1997.

[22] Ramon Sanguesa. *Learning Possibilistic Networks from Data*. PhD thesis, Departament de Llenguatges i Sistemes Informatics, Universitat Politecnica de Catalunya, 1997.

[23] Arno Siebes and Martin L. Kersten. Keso: Minimizing database interaction. In *Proceedings 3rd Conference in Knowledge Discovery in Databases - KDD'97 (to appear)*, 1997.

[24] Milan Studený. Formal properties of conditional independence in different calculi of ai. In *Proceedings European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 341–348. Springer-Verlag, 1993.

[25] Verma T.S. and J. Pearl. The logic of representing dependencies by directed graphs. In *Proceedings AAAI Conference, Seattle, WA.*, pages 374–379, 1987.