Possibilistic Networks: Data Mining Applications

Rudolf Kruse and Christian Borgelt

Dept. of Knowledge Processing and Language Engineering
Otto-von-Guericke University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany

e-mail: {kruse,borgelt}@iws.cs.uni-magdeburg.de

Abstract: The explosion of data stored in commercial or administrational databases calls for intelligent techniques to discover the patterns hidden in them and thus to exploit all available information. Therefore a new line of research has recently been established, which became known under the names "Data Mining" and "Knowledge Discovery in Databases". In this paper we study a popular technique from its arsenal of methods to do dependency analysis, namely learning inference networks (also called "graphical models") from data. We review the already well-known probabilistic networks and provide an introduction to the recently developed and closely related possibilistic networks. The latter can be expected to have some impact on industrial applications, because they are especially suited to handle not only uncertain, but also imprecise information.

1 Introduction

Due to the advances in hardware and software technology, large databases (product databases, customer databases, etc.) are nowadays maintained in almost every company and scientific or administrational institution. But often the data is only recorded; evaluation is restricted to simple retrieval and aggregation operations that can be carried out e.g. by SQL queries. It is obvious that such operations cannot discover broader structures or general patterns that are present in the data. This, obviously, is a waste of information, since knowing such patterns can give a company a decisive competitive edge. Therefore from recent research a new area called "Data Mining" has emerged, which aims at finding "knowledge nuggets" that are hidden in huge volumes of data. It is the operational core of a process called "Knowledge Discovery in Databases", which (in addition to data mining) comprises data selection, data preprocessing, data transformation, visualization, and result evaluation and documentation [6].

Data mining itself can be characterized best by a set of tasks like classification, clustering (segmentation), prediction, etc. In this paper we focus on dependency analysis, i.e. the task to find dependencies between the attributes that are used to describe a domain of interest. A popular method for this task is the automatic induction of inference networks, also called "graphical models", from a set of sample cases.

Graphical models are best known in their probabilistic version, i.e. as Bayesian networks [18] or Markov networks [16]. Efficient implementations of inference systems based on them include HUGIN [1] and PATHFINDER [12]. Probabilistic graphical models are learned from data by searching for the most appropriate decomposition of the multivariate probability distribution induced by a given dataset [5, 13].

Unfortunately probabilistic graphical models suffer from severe difficulties to deal with imprecise, i.e. set-valued, information in the database to learn from. However, the incorporation of imprecise information is more and more recognized as being indispensable for industrial practice. Therefore graphical models are studied also with respect to other uncertainty calculi, either based on a generalization of the modeling technique to so-called valuation-based networks [23, 24], implemented e.g. in PULCINELLA [22], or based on a specific derivation of possibilistic networks, implemented e.g. in POSSINFER [10, 15]. Recently learning possibilistic networks from data has also been studied [9, 11, 2, 3].

Since the approach to try possibilistic or fuzzy methods, if a mechanism for handling uncertainty and imprecision is needed, has been used several times in the past and turned out to be successful in several industrial projects, it can be expected that the same holds true for possibilistic graphical models. Indeed, the application we describe in this paper provides strong evidence for such an expectation.

2 Probabilistic Networks

To describe an object or a case taken from a given domain of interest usually a set of attributes is used, e.g. to describe a car we may consider the manufac-

turer, the model (year), the colour, etc. Depending on the object or case these attributes assume certain values, e.g. VW, Golf, red, etc. In probability theory such attributes are seen as *random variables* which assume their values depending on the *(simple) event* from the *sample space*. Of course, some combinations of attribute values are more frequent than others, e.g. red VW Golf are more frequent than yellow BMW Z1. This frequency information is modeled as a joint probability distribution on the Cartesian product of the attribute domains, i.e. to each combination of values a probability is assigned. However, this joint probability distribution cannot be represented directly, since usually a large number of attributes is necessary to describe an object or a case in sufficient detail, but even a modest number of attributes leads to a huge space of possible combinations of attribute values. Therefore ways to compress the representation are searched for. Among these are the (closely related) Bayesian networks [18] and Markov networks [16].

Bayesian networks, to which we restrict our discussion of probabilistic networks, are based on the product theorem of probability theory. This theorem states that a strictly positive joint probability distribution of a set of random variables can be decomposed into a product of conditional probability distributions. For discrete random variables, to which we confine in the following (we actually assume, that all domains are finite), this theorem reads ($A_1, \ldots, A_n$ are the attributes, $\mathrm{dom}(A_k)$, $k = 1, \ldots, n$, their domains):

$$
\begin{aligned}
&\forall a_{i_1}^{(1)} \in \mathrm{dom}(A_1), \ldots, a_{i_n}^{(n)} \in \mathrm{dom}(A_n): \\
&P\left(A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)}\right) \\
&\quad = P\left(A_n = a_{i_n}^{(n)} \middle| A_{n-1} = a_{i_{n-1}}^{(n-1)}, \ldots, A_1 = a_{i_1}^{(1)}\right) \\
&\qquad \cdot \ \ldots \\
&\qquad \cdot \ P\left(A_3 = a_{i_3}^{(3)} \middle| A_2 = a_{i_2}^{(2)}, A_1 = a_{i_1}^{(1)}\right) \\
&\qquad \cdot \ P\left(A_2 = a_{i_2}^{(2)} \middle| A_1 = a_{i_1}^{(1)}\right) \\
&\qquad \cdot \ P\left(A_1 = a_{i_1}^{(1)}\right).
\end{aligned}
$$

Unfortunately, simply applying the product theorem does not lead to a better representation, but rather to a worse one, since even the conditional distribution needed for the first factor is just as large as the joint distribution itself. However, if for the domain of interest we know a set of conditional independencies

$$
\begin{aligned}
&\forall a_{i_1}^{(1)} \in \mathrm{dom}(A_1), \ldots, a_{i_k}^{(k)} \in \mathrm{dom}(A_k): \\
&P\left(A_k = a_{i_k}^{(k)} \middle| A_{k-1} = a_{i_{k-1}}^{(k-1)}, \ldots, A_1 = a_{i_1}^{(1)}\right) \\
&\quad = P\left(A_k = a_{i_k}^{(k)} \middle| \textstyle\bigcap_{A_j \in \mathrm{parents}(A_k)} A_j = a_{i_j}^{(j)}\right),
\end{aligned}
$$

where $\mathrm{parents}(A_k) \subseteq \{A_1, \ldots, A_{k-1}\}$, it may be possible to simplify the products significantly (provided the sets $\mathrm{parents}(A_k)$ of conditioning attributes are small). These equations are called "conditional independence statements", since they state that given the values of the attributes in $\mathrm{parents}(A_k)$, the attribute A_k is independent of the remaining attributes in $\{A_1, \ldots, A_k\}$. It is obvious that the achievable simplification depends on the order of the attributes when applying the product theorem, since this order determines which conditional independencies can be exploited in the first place. If a bad order is chosen, no simplification may be possible, whereas a good order may lead to considerably smaller sets of conditions.

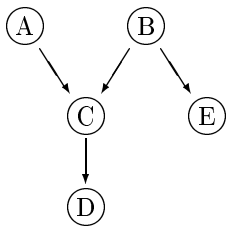


Figure 1: A simple Bayesian network representing the joint probability distribution on the variables $A, \ldots, E$ using the products $\forall a \in \mathrm{dom}(A), \ldots, e \in \mathrm{dom}(E)$: $P(A = a, \ldots, E = e) = P(E = e|B = b) \cdot P(D = d|C = c) \cdot P(C = c|B = b, A = a) \cdot P(B) \cdot P(A)$.

The resulting simplified products are usually represented as a directed graph — this is the reason for the name "graphical model" —, in which there is an edge from each conditioning attribute to the corresponding conditioned attribute, i.e. from each element of $\mathrm{parents}(A_k)$ to A_k, $k = 1, \ldots, n$. This also explains the name $\mathrm{parents}(A_k)$, since the elements of this set thus turn out to be the parent nodes of the attribute A_k in a directed graph (cf. figure 1). To each node the conditional probability distributions of the corresponding attribute given its parent attributes is assigned.

With the help of such a graph inferences can be drawn by propagating observations, i.e. restrictions on the possible values of some attributes, along the edges of the graph — using the conditional probability distributions to update the (marginal) probabilities.

Learning a Bayesian network from data consists in decomposing (factorizing) a given multi-variate probability distribution into products as simple as possible using the means described above. Equivalently we may say that we try to find a dependency graph as sparse as possible. Of course, the distribution to decompose is not given directly, but we are given only a database of sample cases. From this database (conditional) relative frequencies are determined, which are then used to estimate the (conditional) probabilities.

An algorithm to learn a Bayesian network from data always consists of two parts: an evaluation measure and a search method. With the aid of the former a

given decomposition (a given dependency graph) is evaluated, whereas the latter governs which decompositions (which graphs) are considered in the search. Often the evaluation measure can be used to guide the search, since it is usually the goal to maximize (or minimize) its value. There is a large variety of evaluation measures, some of which have been developed especially for learning Bayesian networks [5, 13], while others have been transferred from the closely related task of inducing decision trees [4, 19, 20]. Unfortunately reasons of space prevent us from discussing these measures in detail. An interested reader may take a look at [2, 3]. Of course, there are also several possible search methods, like greedy parent search, optimum weight spanning tree construction, simulated annealing, genetic programming, etc.

3 Possibilistic Networks

The development of possibilistic networks was triggered by the fact that probabilistic networks are well suited to represent and process *uncertain* information, but cannot that easily be extended to handle *imprecise* information. Since the explicit treatment of imprecise information is more and more claimed to be necessary for industrial practice, it is reasonable to investigate graphical models related to alternative uncertainty calculi, e.g. possibility theory.

Maybe the best way to explain the difference between uncertain and imprecise information is to consider the notion of a degree of possibility. The interpretation we prefer is based on the context model [8, 15]. In this model possibility distributions are seen as information-compressed representations of (not necessarily nested) random sets and a degree of possibility as the one-point coverage of a random set [17].

To be more precise: Let ω_0 be the actual, but unknown state of a domain of interest, which is contained in a set Ω of possible states. Let $(C, 2^C, P)$, $C = \{c_1, c_2, \ldots, c_m\}$, be a finite probability space and $\gamma : C \to 2^\Omega$ a set-valued mapping. C is seen as a set of contexts that have to be distinguished for a set-valued specification of ω_0. The contexts are supposed to describe different physical and observation-related frame conditions. $P(\{c\})$ is the (subjective) probability of the (occurrence or selection of the) context c.

A set $\gamma(c)$ is assumed to be the *most specific correct set-valued specification* of ω_0, which is implied by the frame conditions that characterize the context c. By 'most specific set-valued specification' we mean that $\omega_0 \in \gamma(c)$ is guaranteed to be true for $\gamma(c)$, but is not guaranteed for any proper subset of $\gamma(c)$. The resulting *random set* $\Gamma = (\gamma, P)$ is an imperfect (i.e. imprecise *and* uncertain) specification of ω_0. Let π_Γ denote the *one-point coverage of* Γ (the *possibility distribution induced by* Γ), which is defined as

$$\pi_\Gamma : \Omega \to [0,1], \pi_\Gamma(\omega) = P\left(\{c \in C \mid \omega \in \gamma(c)\}\right).$$

In a complete modeling the contexts in C must be specified in detail, so that the relationships between all contexts c_j and their corresponding specifications $\gamma(c_j)$ are made explicit. But if the contexts are unknown or ignored, then $\pi_\Gamma(\omega)$ is the total mass of all contexts c that provide a specification $\gamma(c)$ in which ω_0 is contained, and this quantifies the *possibility of truth* of the statement "$\omega = \omega_0$" [7, 8, 10].

That in this interpretation a possibility distribution represents uncertain *and* imprecise knowledge can be understood best by comparing it to a probability distribution and to a relation. A probability distribution covers *uncertain*, but *precise* knowledge. This becomes obvious, if one notices that a possibility distribution in the interpretation described above reduces to a probability distribution, if $\forall c_j \in C : |\gamma(c_j)| = 1$, i.e. if for all contexts the specification of ω_0 is precise. On the other hand, a relation represents *imprecise*, but *certain* knowledge about dependencies between attributes. Thus, not surprisingly, a relation can also be seen as a special case of a possibility distribution, namely if there is only one context. Hence the context-dependent specifications are responsible for the imprecision, the contexts for the uncertainty in the imperfect knowledge expressed by a possibility distribution.

As a concept of independence we use *possibilistic non-interactivity*. Let X, Y, and Z be three disjoint subsets of variables. Then X is called *conditionally independent* of Y given Z w.r.t. π, if $\forall \omega \in \Omega$:

$$\pi(\omega_{X \cup Y} \mid \omega_Z) = \min\{\pi(\omega_X \mid \omega_Z), \pi(\omega_Y \mid \omega_Z)\}$$

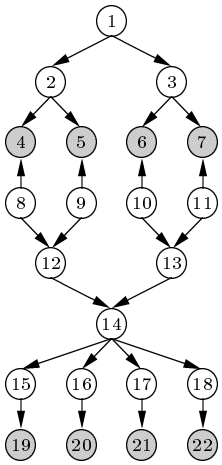
whenever $\pi(\omega_Z) > 0$, where $\pi(\cdot \mid \cdot)$ is a non-normalized conditional possibility distribution

$$\pi(\omega_X \mid \omega_Z) = \max\{\pi(\omega') \mid \omega' \in \Omega \wedge \mathrm{proj}_X(\omega) = \omega_X \wedge \mathrm{proj}_Z(\omega) = \omega_Z\},$$

with $\mathrm{proj}_X(\omega)$ the projection of a tuple ω to the variables in X.

With these ingredients the theory of possibilistic networks can be developed in analogy to the probabilistic case. The only difference is that instead of the product to decompose (factorize) the given joint distribution and the sum to determine the marginal distributions the operations minimum and maximum have to be used: A *possibilistic network* is a decomposition of a multi-variate possibility distribution according to

$$\forall a^{(1)}_{i_1} \in \mathrm{dom}(A_1), \ldots, a^{(k)}_{i_k} \in \mathrm{dom}(A_k) :$$
$$\pi\left(A_1 = a^{(1)}_{i_1}, \ldots, A_n = a^n_{i_n}\right)$$
$$= \min_{i=1}^{n} \pi\left(A_k = a^{(k)}_{i_k} \,\middle|\, \textstyle\bigcap_{A_j \in \mathrm{parents}(A_k)} A_j = a^{(j)}_{i_j}\right),$$



1 – parental error
2 – dam correct?
3 – sire correct?
4 – stated dam ph.gr. 1
5 – stated dam ph.gr. 2
6 – stated sire ph.gr. 1
7 – stated sire ph.gr. 2
8 – true dam ph.gr. 1
9 – true dam ph.gr. 2
10 – true sire ph.gr. 1
11 – true sire ph.gr. 2
12 – offspring ph.gr. 1
13 – offspring ph.gr. 2
14 – offspring genotype
15 – factor 40
16 – factor 41
17 – factor 42
18 – factor 43
19 – lysis 40
20 – lysis 41
21 – lysis 42
22 – lysis 43

The grey nodes correspond to observable attributes. Node 1 can be removed to simplify constructing the clique tree for propagation.

Figure 2: Domain expert designed network for the Danish Jersey cattle blood type determination example.

n	y	y	f1	v2	f1	v2	f1	v2	f1	v2	v2	v2	v2v2	n	y	n	y	0	6	0	6
n	y	y	f1	v2	**	**	f1	v2	**	**	**	**	f1v2	y	y	n	y	7	6	0	7
n	y	y	f1	v2	f1	f1	f1	v2	f1	f1	f1	f1	f1f1	y	y	n	n	7	7	0	0
n	y	y	f1	v2	f1	f1	f1	v2	f1	f1	f1	f1	f1f1	y	y	n	n	7	7	0	0
n	y	y	f1	v2	f1	v1	f1	v2	f1	v1	v2	f1	f1v2	y	y	n	y	7	7	0	7
n	y	y	f1	f1	**	**	f1	f1	**	**	f1	f1	f1f1	y	y	n	n	6	6	0	0
n	y	y	f1	v1	**	**	f1	v1	**	**	v1	v2	v1v2	n	y	y	y	0	5	4	5
n	y	y	f1	v2	f1	v1	f1	v2	f1	v1	f1	v1	f1v1	y	y	y	y	7	7	6	7
...																					

Table 1: An extract from the Danish Jersey cattle blood group determination database.

where parents(A_k) is the set of parents of variable A_k, which is made as small as possible by exploiting conditional independencies of the type indicated above. Just as a Bayesian network it can be represented as a directed graph in which there is an edge from each of the conditioning variables (the parents) to the conditioned variable (the child).

Learning possibilistic networks from data has been studied in [9, 11, 2, 3]. Basically it can follow the same lines as learning probabilistic networks. Again an algorithm for this task has two parts: an evaluation measure and a search method. The latter can be the same as for learning a probabilistic network, but, of course, different evaluation measures have to be used to estimate the network quality. For reasons of space we must skip the details here. An interested reader may consult [11, 2, 3].

4 Applications

As a test case we chose the Danish Jersey cattle blood group determination problem [21], for which a Bayesian network designed by domain experts (cf. figure 2) and a database of 500 real world sample cases exists (an extract of this database is shown in table 1). The problem with this database is that it contains a pretty large number of unknown values — only a little over half of the tuples are complete (This can already be guessed from the extract shown in table 1: the stars denote missing values).

As already indicated above, missing values make it difficult to learn a Bayesian network, since an unknown value can be seen as representing imprecise information: It states that all values contained in the domain of the corresponding attribute are possible. Nevertheless it is still feasible to learn a Bayesian network from the database in this case, since the dependencies are rather strong and thus the remaining small number of tuples is still sufficient to recover the underlying structure. However, learning a possibilistic network from the same dataset is much easier, since possibility theory was especially designed to handle imprecise information. Hence no discarding or special treatment of tuples is necessary. An evaluation of the learned network showed that it was of comparable quality. Thus we can conclude that learning possibilistic networks from data is an important alternative to the established probabilistic methods.

5 Conclusions

In this paper we reviewed, although briefly, the ideas underlying probabilistic networks and provided an equally brief introduction to possibilistic networks. The main advantage of the latter over the former is that they can handle directly imprecise, i.e. set-valued, information. This is especially useful, if an inference network is to be learned from data and the database

to learn from contains a considerable amount of missing values. Whereas in order to learn a probabilistic network these tuples have to be discarded or treated in some complicated manner, possibilistic network learning can easily take them into account and can thus, without problem, make use of all available information. These considerations proved to be well-founded in an application on a real-world database and therefore it can be expected that data mining with possibilistic networks will soon find its way to broader industrial application.

References

[1] S.K. Andersen, K.G. Olesen, F.V. Jensen, and F. Jensen. HUGIN — A shell for building Bayesian belief universes for expert systems. *Proc. 11th Int. J. Conf. on Artificial Intelligence*, 1080–1085, 1989

[2] C. Borgelt and R. Kruse. Evaluation Measures for Learning Probabilistic and Possibilistic Networks. *Proc. 6th IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'97)*, pp. 669–676, Barcelona, Spain, 1997

[3] C. Borgelt and R. Kruse. Some Experimental Results on Learning Probabilistic and Possibilistic Networks with Different Evaluation Measures. *Proc. 1st Int. J. Conf. on Qualitative and Quantitative Practical Reasoning, ECSQARU-FAPR'97*, 71–85, Bad Honnef, Germany, 1997

[4] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*, Wadsworth International, Belmont, CA, 1984

[5] G.F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning* 9:309–347, Kluwer, 1992

[6] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, eds. *Advances in Knowledge Discovery and Data Mining*. AAAI Press / MIT Press, Cambridge, MA, 1996

[7] J. Gebhardt and R. Kruse. A Possibilistic Interpretation of Fuzzy Sets in the Context Model. *Proc. IEEE Int. Conf. on Fuzzy Systems*, 1089-1096, San Diego 1992.

[8] J. Gebhardt and R. Kruse. The context model — an integrating view of vagueness and uncertainty. *Int. Journal of Approximate Reasoning* **9** 283–314

[9] J. Gebhardt and R. Kruse. Learning Possibilistic Networks from Data. *Proc. 5th Int. Workshop on Artificial Intelligence and Statistics*, 233–244, Fort Lauderdale, 1995

[10] J. Gebhardt and R. Kruse. POSSINFER — A Software Tool for Possibilistic Inference. In: D. Dubois, H. Prade, and R. Yager, eds. *Fuzzy Set Methods in Information Engineering: A Guided Tour of Applications*, 407–418, Wiley, New York, 1996

[11] J. Gebhardt and R. Kruse. Tightest Hypertree Decompositions of Multivariate Possibility Distributions. *Proc. Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems*, 1996

[12] D. Heckerman. *Probabilistic Similarity Networks*. MIT Press 1991

[13] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning* 20:197–243, Kluwer, 1995

[14] R. Kruse, E. Schwecke, and J. Heinsohn. *Uncertainty and Vagueness in Knowledge-based Systems: Numerical Methods*. Series: Artificial Intelligence, Springer, Berlin 1991

[15] R. Kruse, J. Gebhardt, and F. Klawonn. *Foundations of Fuzzy Systems*, John Wiley & Sons, Chichester, England 1994

[16] S.L. Lauritzen and D.J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society, Series B*, 2(50):157–224, 1988

[17] H.T. Nguyen. Using Random Sets. *Information Science* 34:265–274, 1984

[18] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (2nd edition)*. Morgan Kaufman, New York 1992

[19] J.R. Quinlan. Induction of Decision Trees. *Machine Learning* 1:81–106, 1986

[20] J.R. Quinlan. *C4.5: Programs for Machine Learning*, Morgan Kaufman, 1993

[21] L.K. Rasmussen. *Blood Group Determination of Danish Jersey Cattle in the F-blood Group System*. Dina Research Report no. 8, 1992

[22] A. Saffiotti and E. Umkehrer. PULCINELLA: A General Tool for Propagating Uncertainty in Valuation Networks. *Proc. 7th Conf. on Uncertainty in AI*, 323–331, San Mateo 1991

[23] G. Shafer and P.P. Shenoy. Local Computations in Hypertrees. Working Paper 201, School of Business, University of Kansas, Lawrence 1988

[24] P.P. Shenoy. Valuation-based Systems: A Framework for Managing Uncertainty in Expert Systems. Working Paper 226, School of Business, University of Kansas, Lawrence, 1991