Possibilistic Networks with Local Structure

Christian Borgelt and Rudolf Kruse

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

e-mail: borgelt@iws.cs.uni-magdeburg.de

Abstract: A recent topic in probabilistic network learning is to exploit local network structure, i.e. to capture regularities in the conditional probability distributions, and to learn networks with local structure from data. In this paper we apply this idea to possibilistic networks, i.e. we try to capture regularities in conditional possibility distributions, and present a modification of the learning algorithm for Bayesian networks with local structure suggested in [7]. The idea underlying this modification is to exploit the decision graph structure that is used to represent the regularities not only to capture a larger set of regularities than decision trees can, but also to improve the learning process.

1 Introduction

Probabilistic inference networks — Bayesian networks [19], but also Markov networks [17] — are well-known tools for reasoning under uncertainty in multidimensional spaces. The idea underlying them is to exploit independence relations between variables in order to decompose a multivariate probability distribution into a set of (conditional or marginal) distributions on lower-dimensional subspaces. Efficient implementations include HUGIN [1] and PATHFINDER [13].

Such independence relations have been studied extensively in the field of graphical modeling [15] and though using them to facilitate reasoning in multidimensional domains has originated in probabilistic reasoning, this approach has been generalized to other uncertainty calculi [23], e.g. in the so-called valuation-based networks [24], and has been implemented e.g. in PULCINELLA [22].

Due to their connection to fuzzy systems and their ability to deal not only with uncertainty but also with imprecision, recently possibilistic networks also gained some attention. They have been implemented e.g. in POSSINFER [10, 16]. In this paper we consider a type of possibilistic network that is based on the context-model interpretation of a degree of possibility and focussed on imprecision [9].

2 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 [9, 16]. 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 [18].

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 "$\omega = \omega_0$" [9, 12].

As a concept of independence, which is fundamental to the technique of graphical modeling, 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.

A *possibilistic network* is a decomposition of a multivariate possibility distribution according to

$$\pi(A_1, \ldots, A_n) = \min_{i=1}^{n} \pi(A_i \mid \mathrm{parents}(A_i)),$$

where $\mathrm{parents}(A_k)$ 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. Such an network is usually represented as a directed graph in which there is an edge from each of the parents to the conditioned variable.

Learning possibilistic networks from data has been studied in [11, 12, 2, 3]. The idea to exploit local structure, which up to now has only been considered for probabilistic networks, can be applied directly to (conditional) possibility distributions, since it is not bound to any specific uncertainty or imprecision calculus.

3 Local Network Structure

Whereas the global structure of a probabilistic or possibilistic network is the directed acyclic graph that encodes the conditional independence statements that hold in a certain domain of interest, the term "local structure" refers to regularities in the conditional probability or possibility tables that are stored with the nodes of the network. Several approaches to exploit such regularities have been studied for Bayesian networks in order to capture additional (i.e. context specific) independencies and (potentially) enhance inference. In this paper we focus on the decision tree/decision graph approach [4, 7] and review it transferred to possibilistic networks.

A very simple way to encode a conditional possibility distribution is a table which for each combination of values of the conditioning variables contains a line stating the corresponding possibility distribution for the values of the conditioned variable. As a simple example, let us consider the small section of a network shown in figure 1 (and let us assume that in this network the variable C has no other parents than variables A and B). Let $\mathrm{dom}(A) = \{a_1, a_2, a_3\}$, $\mathrm{dom}(B) = \{b_1, b_2\}$, and $\mathrm{dom}(C) = \{c_1, c_2\}$. Then the conditional possibilities $\pi(C = c_k \mid A = a_i, B = b_j)$ have to be stored with the node for variable C, e.g. as shown in table 1 (a).

The same set of conditional possibility distributions can also be stored in a tree, in which the leaves hold the conditional possibility distributions and each level of inner nodes corresponds to one conditioning variable (see figure 2). The branches in this tree are labeled with the values of the conditioning variables and thus each path from the root to a leaf corresponds

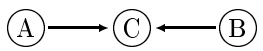

Figure 1: A small section of a possibilistic network.

parents		(a) child		(b) child	
A	B	$C = c_1$	$C = c_2$	$C = c_1$	$C = c_2$
a_1	b_1	π_{11}	π_{12}	π_{11}	π_{12}
a_1	b_2	π_{21}	π_{22}	π_{11}	π_{12}
a_2	b_1	π_{31}	π_{32}	π_{31}	π_{32}
a_2	b_2	π_{41}	π_{42}	π_{41}	π_{42}
a_3	b_1	π_{51}	π_{52}	π_{21}	π_{22}
a_3	b_2	π_{61}	π_{62}	π_{21}	π_{22}

Table 1: Two conditional possibility tables for the section of a possibilistic network shown in figure 1, (a) without, (b) with some regularities.

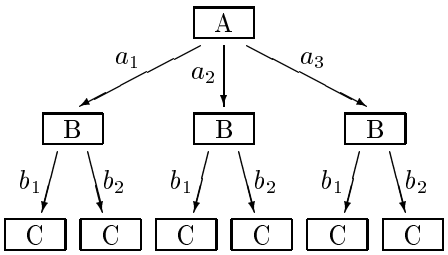

Figure 2: A full decision tree for the variable C.

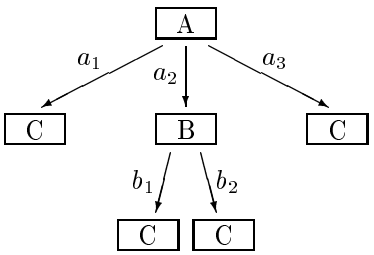

Figure 3: A partial decision tree for the variable C.

parents		(a) child		(b) child	
A	B	$C = c_1$	$C = c_2$	$C = c_1$	$C = c_2$
a_1	b_1	π_{11}	π_{12}	π_{11}	π_{12}
a_1	b_2	π_{11}	π_{12}	π_{11}	π_{12}
a_2	b_1	π_{21}	π_{22}	π_{21}	π_{22}
a_2	b_2	π_{31}	π_{32}	π_{31}	π_{32}
a_3	b_1	π_{21}	π_{22}	π_{31}	π_{32}
a_3	b_2	π_{41}	π_{42}	π_{41}	π_{42}

Table 2: Two conditional possibility tables for the section of a Bayesian network shown in figure 1, with different kinds of regularities.

to one combination of values of the conditioning variables. Obviously such a tree is similar to a decision tree for the variable C (like one learned e.g. by C4.5 [20]) with the following restrictions: All leaves have to lie on the same level and in one level of the tree the same variable has to be tested on all paths. If these restrictions hold, we call the tree a *full* decision tree.

Let us now assume that there are some regularities in the conditional possibility distribution (see table 1 (b)). Since the table clearly shows that the value of the variable B is important only, if A has the value a_2, the tests of variable B can be removed from the branches for the values a_1 and a_3 (see figure 3).

However, a decision tree is not powerful enough to capture all possible regularities. Although we can achieve a lot by tolerating a change in the test order of the variables and by accepting binary splits and multiple tests of the same variable (then, for example, the regularities in table 2 (a) can be represented by a decision tree, which, for reasons of space, is not shown) the regularities shown in table 2 (b) cannot be represented by a decision tree.

The problem is that in a decision tree a test of a variable splits the lines of a conditional possibility table into disjoint subsets that cannot be brought together again. In table 2 (b) a test of variable B thus separates lines 1 and 2 and a test of variable A separates lines 4 and 5. Hence either test prevents us from exploiting one of the two equivalences of possibilities. This drawback can be overcome by allowing a node of the tree to have more than one parent, thus going from decision trees to decision graphs [7]. With decision graphs the regularities in table 2 can easily be captured, see figure 4 (a).

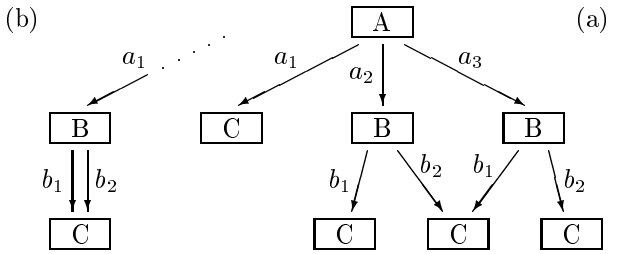

Figure 4: (a) A decision graph for which no equivalent decision tree exists. It captures the regularities in table 2 (b). Replacing the leftmost branch as indicated in (b) turns it into a decision graph with a full set of inner nodes. Note that in this case the test of variable B in the leftmost node on the second level is without effect, since both edges lead to the same leaf.

4 Learning Local Structure

To learn a decision graph from data one can go about as for learning a decision tree. That is, one can define split operations (a full split and a binary split) and in addition (since we are dealing with decision graphs) a merge operation for leaf nodes. The operation to perform is chosen by a greedy approach w.r.t. some evaluation measure. This is, in a nutshell, the approach taken in [7].

Our own approach is a slight modification of the above. The additional degree of freedom of decision graphs compared to decision trees, namely that a node in a decision graph can have more than one parent, can be exploited not only to capture a larger set of regularities but also to improve the learning process for the local structure of a possibilistic network. Our idea is as follows: With decision graphs, we can always work with the complete set of inner nodes of a full decision tree and let only leaves have more than one parent. Even if we do not care about the order of the conditioning variables in the decision structure and allow only one test per variable on each path, such a structure can capture all regularities of the examples examined in the preceding section. E.g. the regularities of table 2 are captured by the decision graph with a full set of inner nodes shown in figure 4 (b).

It is easy to see that such an approach can capture any regularities that may be present in conditional possibility tables. Basically, merging the leaves of a full decision tree is the same as merging arbitrary lines of a conditional possibility table. The decision graph structure just makes it much easier to keep track of the different value combinations of the conditioning (i.e. parent) variables, for which the same possibility distribution for the values of the conditioned (i.e. child) variable has to be adopted.

In a learning algorithm we use only two operations, namely (1) adding a new level to a decision graph, i.e. splitting all leaves according to the values of a new parent variable, and (2) merging two leaves into one. Which leaves to merge is determined, just as above, in a greedy fashion w.r.t. some evaluation measure. The first step (adding a new level), which may seem to be costly, does no harm, since it is necessary, even if one only learns a possibilistic network without local

structure (provided the conditional distributions are represented as a decision tree). Only this step involves a visit to the database to learn from in order to determine the conditional value frequencies. The next step, in which leaves are merged, can be carried out without visiting the database, since all necessary information is already available in the leaf nodes (provided the original leaf nodes are kept during a trial merge and are simply restored afterwards). Thus we need to visit the database only as often as an algorithm for learning a possibilistic network without local structure does.

In contrast to this, the algorithm presented in [7] needs to visit the database each time a split of leaf nodes is considered. This can exceed by far the number of times an algorithm for learning a network without local structure needs to visit the database, especially, since multiple tests of the same variable along the same path are permitted.

Of course, our approach can result in a complicated structure that may hide a simple structure of context-specific independencies. But the same is true, though maybe less likely, for the algorithm presented in [7] and thus some postprocessing to simplify the structure by changing the order of the variables and by splitting tests along a path is always advisable.

eval. measure	num. of conds.	num. of params.	network quality
indep. vars.	0	84	158.2
poss. χ^2	35	1074	141.7
mut. spec.	33	778	144.7
S_{gain}	31	1018	143.0
S_{gr}	18	200	154.2
S_{sgr}	28	436	149.2

Table 3: Results of possibilistic network learning without local structure.

eval. measure	num. of conds.	num. of params.	network quality
poss. χ^2	35	1068	141.7
mut. spec.	33	708	142.0
S_{gain}	31	795	145.9
S_{gr}	24	157	157.5
S_{sgr}	30	494	152.6

Table 4: Results of possibilistic network learning with local structure.

5 Experimental Results

All experiments reported here were carried out with a prototype learning program for probabilistic and possibilistic networks called INES (Induction of NEtwork Structures, written by the first author of this paper), into which the described method is incorporated. This program also lets you choose from a variety of evaluation measures described in [2]. As a test case we chose the Danish Jersey cattle blood group determination problem [21], for which a Bayesian network designed by domain experts and a database of 500 real world sample cases exist.

To evaluate the quality of the learned network, we chose the following approach: Given a possibilistic network, the possibility degree of any (complete) tuple can be computed. If a tuple contains missing values, we assign to this tuple the maximal possibility degree over all complete tuples that are compatible with this tuple. The sum of these possibility degrees we used as a quality measure. This is justified, since due to the the way in which a possibilistic network approximates a multivariate possibility distribution, the possibility degree resulting from the network must always be equal or greater than the true possibility degree. Hence, the lower the sum of the possibility degrees for the tuples in the database, the better the network. More details about this evaluation method can be found in [3].

The results of some of our experiments are shown in table 4. In addition to the network evaluation, these tables show the total number of conditions (parents) as a measure of the complexity of the global network structure and the number of parameters as a measure of the complexity of the local network structure.

At first sight it is surprising that allowing local structure to be learned, although it often leads to a reduction of the number of necessary parameters, sometimes makes the global structure more complex (i.e. leads to a larger number of conditions as for networks without local structure). But a second thought (and a closer inspection of the learned networks) reveals, that this could have been foreseen. In a frequency distribution determined from a database of sample cases random fluctuations are to be expected. Usually these do not lead to additional conditions, since the evaluation measures prevent a selection of too many parents, simply because they "dislike" a lot of (roughly equivalent) leaves. But the disadvantage of (approximately) equivalent leaves is removed by the possibility to merge these leaves, and thus those fluctuations that show a higher deviation from the true (independent) distribution are filtered out and become significant to the measures.

This effect reduces when learning from a larger dataset, but does not vanish completely. We believe this to be a general problem any learning algorithm for local structure has to cope with. Therefore it may be advisable not to combine learning global and local network structure, but to learn the global structure first and to simplify the learned structure afterwards by learning the local structure.

6 Conclusions

In this paper we presented a method to learn the local structure of a possibilistic network from data, which is derived from the approach to learn Bayesian networks with local structure presented in [7]. The experimental results show that trying to learn local structure has to be handled with care, since it can lead to the counter-intuitive effect of a more complicated global structure (though this depends on the evaluation measure). It may be advisable to base selecting another parent on the score for a full decision tree, and to use local structure learning only to simplify this tree afterwards.

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)*, Vol. 2:pp. 1034–1038, 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. Joint Conference on Qualitative and Quantitative Practical Reasoning (ECSQARU/FAPR'97)*, pp. 71–85, Springer, Berlin, Germany, 1997)

[4] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context Specific Independence in Bayesian Networks. *Proc. 12th Conf. on Uncertainty in Artificial Intelligence (UAI'96)*, Portland, OR, 1996

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

[6] W. Buntine. Theory Refinement on Bayesian Networks. *Proc. 7th Conf. on Uncertainty in Artificial Intelligence*, pp. 52–60, Morgan Kaufman, Los Angeles, CA, 1991

[7] D.M. Chickering, D. Heckerman, and C. Meek. A Bayesian Approach to Learning Bayesian Networks with Local Structure. *Proc. 13th Conf. on Uncertainty in Artificial Intelligence (UAI'97)*, pp. 80–89, Morgan Kaufman, San Franscisco, CA, 1997

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

[9] J. Gebhardt and R. Kruse. The context model — an integrating view of vagueness and uncertainty *Int. Journal of Approximate Reasoning* **9** 283–314, 1993

[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*, Wiley 1995

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

[12] 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

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

[14] 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

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

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

[17] 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

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

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

[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