Learning Undirected Possibilistic Networks with Conditional Independence Tests

Christian Borgelt

***Abstract*—Approaches based on conditional independence tests are among the most popular methods for learning graphical models from data. Due to the predominance of Bayesian networks in the field, they are usually developed for directed graphs. For possibilistic networks of a certain kind, however, undirected graphs are a more natural basis and thus algorithms for learning undirected graphs are desirable in this area. In this paper I present an algorithm for learning undirected graphical models, which is derived from the well-known Cheng–Bell–Liu algorithm. Its main advantage is the lower number of conditional independence tests that are needed, while it achieves results of comparable quality.**

I. Introduction

After modeling complex domains with graphical models, and especially Bayesian networks, had become a popular research area in the eighties [19], [24], [17], [9], [15], [12], learning graphical models from data was a focus of attention in the nineties [4], [21], [7], [14], [5], [16], [23], [6], [3], [18]. Several search methods and evaluation measures, which are the core ingredients of any algorithm for learning graphical models, were developed and applied not only to learning probabilistic graphical models, but also to the somewhat less well known possibilistic networks [13], [1], [8], [3].

Highly popular among these algorithms are approaches that are based on conditional independence tests [22], [5], [23], [6], because they promise several advantages over other methods. For example, they do not require any prior information about the domain (the K2 algorithm, for example, requires a topological order of the attributes) and do not restrict the graphs that can be learned. They may even be used to enhance other methods [21]. Unfortunately, though, due to the predominance of Bayesian networks in the research on graphical models, they are usually developed for directed graphs.

However, for a certain type of possibilistic networks [3] undirected graphs are a much more natural basis. Of course, it is always possible to turn a directed graph into a corresponding undirected one by simply “moralizing” the graph (that is, by adding edges between unconnected parents of a node). This may lose independence information, but cannot lead to incorrect models. Nevertheless it is desirable to have methods that can learn undirected graphical models directly. In such algorithms the missing edge directions can be exploited, for example, to avoid certain tests and thus to achieve faster and possibly also more robust learning.

Christian Borgelt is with the European Center for Soft Computing, Edificio Científico-Tecnológico, c/ Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Asturias, Spain (email: christian.borgelt@softcomputing.es).

In this paper I present an algorithm for learning undirected graphical models that is derived from the well-known Cheng–Bell–Liu algorithm [5], [6], but needs fewer conditional independence tests, while it achieves results of equal quality. This paper is organized as follows: in Section II I briefly review some basics of conditional independene tests needed in the algorithm and in Section III recall the Cheng–Bell–Liu algorithm. In Section IV I present my algorithm for learning undirected graphical models. Section V reports experiments which compare the results of the two algorithms and Section VI draws conclusion from the discussion.

II. Conditional Independence Tests

The core of the algorithms, which are at the focus of this paper, is a conditional independence test. Such a test usually consists of an evaluation measure for assessing the strength of the (conditional) dependence of two variables together with a threshold. If the value of the measure is below the threshold, the variables are considered to be (conditionally) independent, otherwise they are judged to be (conditionally) dependent.

There is a large number of possible evaluation measures, especially for probabilistic independence. The reason is that basically all measures used in decision tree induction can be transferred and then yield such a measure. Even if a measure in its standard form is meant only for assessing the strength of marginal dependence, it can usually be extended to yield a measure for conditional dependence by simply computing its value for each possible instantiation of the condition attributes and then aggregating these values in a suitable fashion. For example, one of the most common measures for the strength of marginal (probabilistic) dependence is *information gain*,

$$I_{\text{gain}}(A,B) = \sum_{i,j} \log_2 \frac{p_{ij}}{p_{i.}p_{.j}}$$

(where A and B are two variables, i and j range over their respective values, and p_{ij}, $p_{i.}$ and $p_{.j}$ are the probabilities of the joint and individual occurrence of these values). It can be extended to conditional information gain by simply summing its value over all instantiations of the conditions,

$$I_{\text{gain}}(A,B|C) = \sum_{k} p_{..k} \sum_{i,j} \log_2 \frac{p_{ijk}}{p_{i.k}p_{.jk}}.$$

(Note that here C may be an individual variable or a set of variables; and then k refers to individual values or value vectors, respectively.) Other examples include the well-known χ^2 measure or the Gini index.

There is also a variety of measures for the possibilistic case [1], [3], for example *specificity gain*,

$$S_{\text{gain}}(A,B) = \text{nsp}(\pi_A) + \text{nsp}(\pi_B) - \text{nsp}(\pi_{AB}).$$

$\text{nsp}(\pi)$ is the *nonspecificity* of the possibility distribution π,

$$\text{nsp}(\pi) = \int_0^{\sup_{E\in\mathcal{E}} \pi(E)} \log_2 \Big(\sum_{E\in\mathcal{E}} [\pi]_\alpha(E) \Big)\, d\alpha,$$

where $\mathcal{E}$ is the set of elementary events underlying the possibility distribution π.

Other measures include the following, which are used in the experimental results section: in the first place, specificity gain may be normalized in analogy to (probabilistic) information gain, which serves the purpose to eliminate or at least lessen a possible bias towards many-valued attributes. This leads to the *specificity gain ratio*,

$$\begin{aligned} S_{\text{gr}}(A,B) &= \frac{S_{\text{gain}}(A,B)}{\text{nsp}(\pi_A)} \\ &= \frac{\text{nsp}(\pi_A) + \text{nsp}(\pi_B) - \text{nsp}(\pi_{AB})}{\text{nsp}(\pi_A)} \end{aligned}$$

and two *symmetric specificity gain ratios*, namely

$$\begin{aligned} S_{\text{sgr1}}(A,B) &= \frac{S_{\text{gain}}(A,B)}{\text{nsp}(\pi_{AB})} \qquad \text{and} \\ S_{\text{sgr2}}(A,B) &= \frac{S_{\text{gain}}(A,B)}{\text{nsp}(\pi_A) + \text{nsp}(\pi_B)}. \end{aligned}$$

By exploiting the fact that (probabilistic) information gain can be written in two ways, another possibilistic measure can be derived. It relies on the form that is usually called "mutual information" (even though it is the same measure as information gain, since the formulas are equivalent) and compares the actual joint distribution and an assumed independent distribution by computing their pointwise quotient, that is,

$$d_{\text{mi}}(A,B) = -\sum_{i=1}^{n_A}\sum_{j=1}^{n_B} \pi_{ij} \log_2 \frac{\pi_{ij}}{\min\{\pi_{i.}, \pi_{.j}\}},$$

where π_{ij} denotes the degree of possibility of the combination of the i-th value of attribute A and the j-th value of attribute B. $p_{i.}$ and $p_{j.}$ are the marginal degrees of possibility, which result from taking the maximum over all values of the missing index.

Finally, the χ^2 measure, which is well known in statistics and computes a pointwise squared difference, can be transferred to possibility distributions. Thus we get

$$d_{\chi^2}(A,B) = \sum_{i=1}^{n_A}\sum_{j=1}^{n_B} \frac{(\min\{\pi_{i.}, \pi_{.j}\} - \pi_{ij})^2}{\min\{\pi_{i.}, \pi_{.j}\}}.$$

Other measures, as well as a more extensive discussion of the underlying ideas, can be found in [3].

All of these measures are extended to conditional tests by summing their values over all instantiations of the conditions, weighted with the degree of possibility of the instantiation. This mimics the way in which conditional (probabilistic) information gain is constructed from its marginal form.

A problem of conditional independence tests is their order, that is, the number of condition attributes. Since the data sets that are available in practice are always limited in size, conditional independence tests become quickly unreliable, the higher their order. Hence algorithms for learning graphical models from data must take care to keep the order of conditional independence tests low. For example, the Cheng–Bell–Liu algorithm, which is reviewed in the next section, does so by exploiting an already constructed graphical model to determine suitable condition sets.

III. The Cheng–Bell–Liu Algorithm

The Cheng–Bell–Liu algorithm [5], [6] constructs a graphical model from a data set of sample cases. It assumes that the given domain can accurately be represented by a perfect map, that is, by a directed graph that captures exactly the conditional independences that are present in the probability distribution governing the data generation process. In addition, it assumes that the evaluation measure used for the conditional independence tests is, in a certain sense, well-behaved. Among other things, this means that the result of a conditional independence test coincides with the actual situation. In addition, if a condition that is needed for the conditional independence of two attributes is removed, the value of the measure should increase, regardless of what other attributes are present in the conditions. For more details about the exact assumptions underlying the Cheng–Bell–Liu algorithm, see [6].

The Cheng–Bell–Liu algorithm works in four phases:

A. *Drafting*

A Chow–Liu tree [10] is built as an initial graphical model. This involves evaluating all pairs of attributes with an (unconditional) independence test. The standard form of the algorithm uses information gain as the evaluation measure and a threshold of about 0.1 bits. Attribute pairs having an evaluation lower than the threshold are permanently discarded from the construction process. The rest form a set of candidate edges, which are weighted with the evaluation. An optimum weight spanning tree that can be constructed from these edges yields the initial graphical model.

B. *Thickening*

The candidate edges that were not used for the initial graphical model are traversed in the order of decreasing evaluation. Each of these edges is tested whether it may be needed in the graphical model. The test exploits the current graphical model in order to find a good set of conditions, namely by selecting the set of adjacent nodes of one attribute that lie on paths leading to the other attribute. The rationale underlying this scheme is the local Markov property of directed conditional independence graphs: an attribute is conditionally independent of all non-descendants given its parents. Of course, since the graph is undirected at this point we cannot determine which attribute is a non-descendant of the other, and hence both possibilities may have to be tried (one trial with the neighbors of one attribute, another with the neighbors of the other).

In addition, the set of adjacent nodes may not only include parents, but also children. Children are a problem, because in the true graphical model there may be a v-structure on the path between the nodes. Including such a node in the conditions is harmful as it activates the path and thus hinders conditional independence. Therefore the set of adjacent nodes is reduced iteratively and greedily. In each step the condition, which, if removed, lowers the dependence score most, is discarded from the conditions. This reduction is carried out until the evaluation falls below the given independence threshold or no removal of a condition lowers the score. In the former case, if it occurs in either of the the two trials, the attributes of the tested edge are judged to be conditionally independent and the edge is not added to the graphical model, otherwise it is added.

C. Thinning

In the thickening phase a test whether an edge is needed was based on a graph that was possibly still to sparse to reveal the conditional independence of a pair of attributes. This is the case, because not all paths that exist between the two attributes in the true model may have been present at the time of the test (even though they must all be present at the end of the thickening phase, provided the assumptions underlying the algorithm hold [6]). Hence the edges that are present in the graphical model after the thickening phase are traversed again and it is retested whether they are needed. This test is carried out in two ways: first in the way described on the thickening phase (which is the "heuristic" form of the test). However, there are certain degenerate cases where this test does not correctly identify an existing conditional independence and thus a superfluous edge may not be removed (see [6] for an example). Hence a strict test, which adds the neighbors of the neighbors of an attribute to the condition set (because two adjacent nodes on a path cannot both have v-structures) before this set is reduced is carried out (this is called the "strict" form of the test). If any of these tests reveals that two attributes are conditionally independent, the corresponding edge is removed from the graph. It can be shown that the resulting graph is a skeleton for the perfect map describing the domain (that is, it contains exactly the needed edges, only directions are missing), provided the underlying assumptions hold.

D. Orienting

In the last phase the edges of the graph are directed. This is done in two steps: first the v-structures are identified and then the remaining edges are oriented. (The latter step may be done automatically or manually.) In order to find the v-structures all pairs of attributes with common neighbors are traversed and it is determined by a (strict) conditional independence test (as described above) which common neighbors are in the (maximally) reduced set of conditions. Those that are removed must be common children and hence a v-structure is built with them. Afterwards other edges may be directed by alternatingly extending directed chains and choosing random directions for edges, the direction of which is not fixed by the v-structures and already extended chains.

IV. Learning Undirected Graphs

Of course, one way of constructing an undirected graphical model consists in executing the Cheng–Bell–Liu algorithm and moralizing the resulting graph. However, from the above description of the algorithm it is clear that in this case a lot of unnecessary work is done. For example, edges are directed and then their direction is discarded again when the graph is moralized. In the tests whether an edge is needed, it is tried to remove children from the condition sets even though in an undirected graph there is no concept of child or parent. Hence it is desirable to devise an algorithm that removes this unnecessary work and thus obtains an undirected graphical model faster and maybe also in a more reliable way.

The proposed algorithm works in four phases:

A. Drafting

This phase is identical to the Cheng–Bell–Liu Algorithm, that is, a Chow–Liu tree is formed (optimum weight spanning tree from edge weights that represent dependence strengths).

B. Thickening

As in the Cheng–Bell–Liu algorithm, the remaining candidate edges (edges with an evaluation above the threshold, but not used in the initial graphical model) are traversed and tested. If the test indicates that they may be needed, they are added to the graphical model. The difference to the Cheng–Bell–Liu algorithm consists in how the tests are executed. The rationale is analogous, but uses the local Markov property of undirected graphs: an attribute is conditionally independent of any other attribute given its neighbors. We may even restrict the set of neighbors to those lying on paths leading to the other attribute (even though this is, strictly speaking, already part of the global Markov property). There is no need for any reduction of the condition set, since we need not take care of children with v-structures. In principle, only a single conditional independence test is needed per edge. However, in order to improve the robustness of the algorithm, it may be advisable to carry out the alternative test (that is, using the neighbors of the other attribute) if it fails. If the test indicates that the attributes are *not* conditionally independent given the current graph structure, the edge is added to the graph.

C. Moralizing

If one makes the assumption that there is an undirected perfect map of the domain, the thinning phase (see below) already yields the result. However, this would not be a feasible assumption. Dependence structures that involve (directed) v-structures are much too frequent in practice and unfortunately there is no lossless way of representing v-structures as undirected graphs. Hence making the assumption that there is an undirected perfect map would render the algorithm basically useless for practical purposes. However, in order to take care of v-structures one only has to connect the parents, that is, one has to moralize the graph. The reason is that even though the parents are independent given their common ancestors (which may be the empty set), they become dependent once a common child is given. This non-monotone behavior (enlarging the set of conditions destroys a conditional independence) cannot be expressed in undirected graphs and thus one must allow for an unrepresented conditional independence.

The consequence of these considerations for the algorithm are clear: some of the edges which have an unconditional evaluation below the threshold (and thus were discarded before the initial graphical model was constructed) may be needed in the graphical model in order to achieve monotony w.r.t. conditional independence. However, for these tests only edges need to be considered that have a common neighbor in the graph constructed so far, since no other edges can connect nodes that are involved in a v-structure. Therefore all such edges are traversed and it is tested whether they are needed in the graph. If the corresponding attributes are *not* found to be conditionally independent given the current graph, the edge is added.

D. Thinning

As for the Cheng–Bell–Liu algorithm it holds that the graph resulting from the preceding step contains superfluous edges, since when the test was carried out not all necessary paths may have been present in the graph. Hence all edges of the graph are retested and those found to be unnecessary are removed from the graph.

As a simple extension of this algorithm one may add a second thinning phase between the thickening phase and the moralizing phase. The idea of such a phase is that the graph resulting from the thinning phase may contain fewer attribute pairs with common neighbors, so that fewer tests have to be carried out in the moralizing phase. Furthermore, the order of the conditional independence tests may be lower, because a lower number of edges can generally be expected to reduce the number of neighbors that enter the condition sets.

The additional costs for such a phase are negligible, since the results of already executed conditional independence tests will be stored in a cache anyway to avoid redundant computations. Hence only for edges between attributes that received a new incident edge in the moralizing step a new test has to be carried out in the second thinning phase. The result of all other tests are already present in the cache and thus can be reused basically without costs.

V. Experiments

In order to test the learning algorithm described above I implemented it as part of the INeS program, which also comprises a large number of other learning algorithms for probabilistic and possibilistic graphical models, including the Cheng–Bell–Liu algorithm.

As a test case I chose a standard benchmark, namely the well-known BOBLO (BOvine BLOod) network and data set [20], which is also known as the Danish Jersey cattle data. It consists of a manually constructed Bayesian network over 21 attributes, which serves the purpose to verify parentage for pedigree registration, and a real-world data set of 500 cases, a large number of which contain missing values.

TABLE I
Network quality with Cheng–Bell–Liu Algorithm

measure	edges	params	min.	avg.	max.
S_{gain}	16	678	8.784	8.932	10.620
S_{sgr1}	20	4701	8.638	8.826	10.340
d_{χ^2}	14	9450	9.344	9.446	10.728
d_{mi}	17	1010	8.460	8.538	10.406

TABLE II
Network quality of learning an undirected graphical model

measure	edges	params	min.	avg.	max.
S_{gain}	17	639	8.738	8.888	10.614
S_{sgr1}	21	3847	8.740	8.926	10.464
d_{χ^2}	12	3442	9.352	9.453	10.730
d_{mi}	19	570	8.586	8.656	10.558

I focused on learning a possibilistic network, since at least the kind in which conditional distributions are identified with joint distributions on the same subspace, is more naturally represented with the help of an undirected graph. In the implementation I made use of a method to efficiently compute maximum projections of a multivariate possibility distribution [2], which is represented by a database of sample cases with missing values or set-valued information.

As already pointed out in Section II, I collected evaluation measures from [1], [3], taking the restriction into account that the measure must be symmetric in order to be usable for a conditional independence test. As a consequence I chose the specificity gain S_{gain}, a symmetric ratio of the specificity gain S_{sgr1} (the two variants mentioned above behave similarly), the possibilistic analog of the χ^2 measure d_{χ^2}, and the possibilistic analog of mutual information d_{mi} (see Section II for the definitions). As independence thresholds I used 0.015 for S_{gain}, 0.1 for S_{sgr1} and d_{χ^2}, and 0.04 for d_{mi}. These values were chosen, because they yielded reasonably good results. By tuning these values, the complexity of the learned network may be controlled to some degree: the higher the threshold, the sparser the learned network.

The results of the experiments together with information about the number of conditional independence tests needed are shown in Tables I and II. Table I shows the size and the quality of the networks learned with the Cheng–Bell–Liu algorithm and different evaluation measures. The number of edges (in the moralized graph) and the number of parameters show the considerably different behavior of the measures w.r.t. the complexity of the learned network. The last three columns state the sums of the minimum, average, and maximum possibility degree of points covered by the tuples in the database, which should be as low as possible (see [3] for more details about this way of evaluating learned possibilistic graphical models).

As could also be observed in experiments with other learning algorithms, d_{χ^2} tends to learn large networks. However, they are usually of better quality than the ones built by the Cheng–Bell–Liu algorithm. This could not be improved by increasing the independence threshold: although the networks get smaller when doing so, they also get even worse, thus

TABLE III
NUMBER OF TESTS WITH STRICT CHENG–BELL–LIU ALGORITHM

	marg.	order of cond. tests					
measure	tests	1	2	3	4	5	any
$S_{\rm gain}$	210	24	14	9	2	0	49
$S_{\rm sgr1}$	210	67	50	33	20	3	173
d_{χ^2}	210	19	18	35	12	3	87
$d_{\rm mi}$	210	28	37	26	9	0	100

TABLE IV
NUMBER OF TESTS WITH HEURISTIC CHENG–BELL–LIU ALGORITHM

	marg.	order of cond. tests					
measure	tests	1	2	3	4	5	any
$S_{\rm gain}$	210	24	14	9	2	0	49
$S_{\rm sgr1}$	210	71	48	30	20	3	172
d_{χ^2}	210	19	18	32	11	3	83
$d_{\rm mi}$	210	31	39	14	2	0	86

indicating that this measure may not be well suited for an approach based on conditional independence tests. With the possibilistic analog of mutual information $d_{\rm mi}$ and the specificity gain $S_{\rm gain}$, however, results are obtained that are competitive with other learning approaches. This confirms earlier results [3] that $d_{\rm mi}$ is among the most recommendable measures for learning possibilistic graphical models.

Table II shows the results for the presented algorithm that learns undirected graphical models directly. It can be observed that it tends to learn smaller networks: even though the number of edges is the same or even slightly larger, the number of parameters is clearly smaller for all measures. Nevertheless the quality of the learned networks is comparable, only for $d_{\rm mi}$ the quality deteriorates slightly. However, this may be counteracted by adapting the independence threshold.

My main goal when developing the algorithm was to reduce the number of the conditional independence tests, especially of higher order. Therefore Tables III to V show the number of (conditional) independence tests that were carried out by the algorithms. Since the drafting phase is identical in both algorithms, all algorithms execute the same number of marginal independence test, namely 210. This number obviously cannot be reduced. However, there are considerable differences in the number of conditional independence tests carried out.

In order to treat the original Cheng–Bell–Liu algorithm as fairly as possible, I executed it in two versions. The first version is the one as it was described above and the numbers of tests needed are shown in Table III. The second version in simplified in as far as in the thinning phase it only carries out heuristic tests (using only direct neighbors in the condition sets) and skips the strict tests. Both versions lead to the same networks on this example problem and thus Table I applies for both version. However, in terms of the number of conditional independence tests one can expect fewer and lower order tests with the second version. This is confirmed by Table IV, although the gains are a lot smaller than I expected. The reason may be that for the orienting phase strict tests are needed anyway and thus skipping them in the thinning phase does not help much. The biggest savings result for $d_{\rm mi}$, where the number of tests goes down by 14.

TABLE V
NUMBER OF TESTS WITH ALGORITHM FOR UNDIRECTED GRAPHS

	marg.	order of cond. tests					
measure	tests	1	2	3	4	5	any
$S_{\rm gain}$	210	13	11	5	4	0	33
$S_{\rm sgr1}$	210	42	29	16	3	0	90
d_{χ^2}	210	9	7	2	8	0	26
$d_{\rm mi}$	210	13	15	20	0	0	48

In contrast to these fairly small gains the gains achieved by using the presented algorithm for learning undirected graphs are considerable. The number of conditional independence tests is cut to about half of the tests needed by the Cheng–Bell–Liu algorithm. The biggest gains result for d_{χ^2}, but also $d_{\rm mi}$, the most recommendable measure benefits significantly. It is worth noting that for all measures except $S_{\rm gain}$ the highest order of a conditional independence test drops by 1.

VI. CONCLUSIONS

In this paper I presented an algorithm for learning undirected graphical models from data that is based on conditional independence tests and derived from the well-known Cheng–Bell–Liu algorithm. It is particularly useful for possibilistic networks, since at least for a certain kind of these networks undirected graphs are the more natural representation. The expectation that the algorithm would achieve its results with fewer conditional independence tests or conditional independence tests of lower order was clearly confirmed in the reported experiments. Hence if undirected graphical models are to be learned, the presented algorithm provides significant advantages. It may be conjectured that it is not only faster, but also provides more reliable results, since the number and order of the tests is decisive in this respect. However, this needs to be substantiated with more experiments.

Software

The INeS program, which is written in C and was used to carry out the experiments described above, can be downloaded free of charge at http://www.borgelt.net/ines.html.

REFERENCES

[1] C. Borgelt and R. Kruse. Evaluation Measures for Learning Probabilistic and Possibilistic Networks. *Proc. 6th IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'97, Barcelona, Spain)*, Vol. 2:1034–1038. IEEE Press, Piscataway, NJ, USA 1997

[2] C. Borgelt and R. Kruse. Efficient Maximum Projection of Database-Induced Multivariate Possibility Distributions. *Proc. 7th IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'98, Anchorage, Alaska, USA)*, CD-ROM. IEEE Press, Piscataway, NJ, USA 1998

[3] C. Borgelt and R. Kruse. *Graphical Models — Methods for Data Analysis and Mining.* J. Wiley & Sons, Chichester, United Kingdom 2002

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

[5] J. Cheng, D.A. Bell, and W. Liu. Learning Belief Networks from Data: An Information Theory Based Approach. *Proc. 6th ACM Int. Conf. Information Theory and Knowledge Management (CIKM'97, Las Vegas, NV)*, 325–331. ACM Press, New York, NY, USA 1997

[6] J. Cheng, R. Greiner, J. Kelly, D.A. Bell, and W. Liu. Learning Bayesian Networks from Data: An Information Theory Based Approach. *Artificial Intelligence* 137(1–2):43–90. Elsevier, Amsterdam, Netherlands 2002

[7] W. Buntine. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research* 2:159–225. Morgan Kaufman, San Mateo, CA, USA 1994

[8] L.M. de Campos, J.F. Huete, and S. Moral. Independence in Uncertainty Theories and Its Application to Learning Belief Networks. In: [11], 391–434.

[9] E. Castillo, J.M. Gutierrez, and A.S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, New York, NY, USA 1997

[10] C.K. Chow and C.N. Liu. Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Trans. on Information Theory* 14(3):462–467. IEEE Press, Piscataway, NJ, USA 1968

[11] D. Gabbay and R. Kruse, eds. *DRUMS Handbook on Abduction and Learning*. Kluwer, Dordrecht, Netherlands 2000.

[12] J.A. Gamez, S. Moral, and A. Salmeron. *Advances in Bayesian Networks*. Springer, Berlin, Germany 2004

[13] J. Gebhardt. *Learning from Data: Possibilistic Graphical Models.* Habilitation Thesis, University of Braunschweig, Germany 1997

[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, Dordrecht, Netherlands 1995

[15] F.V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, Berlin, Germany 2001

[16] M.I. Jordan, ed. *Learning in Graphical Models*. MIT Press, Cambridge, MA, USA 1998

[17] S.L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, United Kingdom 1996

[18] R.E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, Englewood Cliffs, NJ, USA 2003

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, USA 1988 (2nd edition 1992)

[20] L.K. Rasmussen. *Blood Group Determination of Danish Jersey Cattle in the F-blood Group System (Dina Research Report 8)*. Dina Foulum, Tjele, Denmark 1992

[21] M. Singh and M. Valtorta. An Algorithm for the Construction of Bayesian Network Structures from Data. *Proc. 9th Conf. on Uncertainty in Artificial Intelligence (UAI'93, Washington, DC, USA)*, 259–265. Morgan Kaufmann, San Mateo, CA, USA 1993

[22] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search (Lecture Notes in Statistics 81)*. Springer, New York, NY, USA 1993

[23] H. Steck. *Constraint-Based Structural Learning in Bayesian Networks using Finite Data Sets*. Ph.D. thesis, TU München, Germany 2001

[24] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. J. Wiley & Sons, Chichester, United Kingdom 1990