Some Experimental Results on Learning Probabilistic and Possibilistic Networks with Different Evaluation Measures

Christian Borgelt and Rudolf Kruse

Department of Information and Communication Systems
Otto-von-Guericke-University of Magdeburg
D-39106 Magdeburg, Germany

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

Abstract. A large part of recent research on probabilistic and possibilistic inference networks has been devoted to learning them from data. In this paper we discuss two search methods and several evaluation measures usable for this task. We consider a scheme for evaluating induced networks and present experimental results obtained from an application of INES (Induction of NEtwork Structures), a prototype implementation of the described methods and measures.

1 Introduction

Since reasoning in multi-dimensional domains tends to be infeasible in the domains as a whole — and the more so, if uncertainty is involved — decomposition techniques, that reduce the reasoning process to computations in lower-dimensional subspaces, have become very popular. For example, decomposition based on dependence and independence relations between variables has extensively been studied in the field of graphical modeling [16]. Some of the best-known approaches are Bayesian networks [22], Markov networks [19], and the more general valuation-based networks [28]. They all led to the development of efficient implementations, for example HUGIN [1], PULCINELLA [27], PATHFINDER [11] and POSSINFER [7].

A large part of recent research has been devoted to learning such inference networks from data [4, 12, 8]. In this paper we examine two search methods and several evaluation measures usable for this task. We consider how to evaluate an induced network and present some experimental results we obtained from an application of INES (Induction of NEtwork Structures), a prototype implementation of the described search methods and evaluation measures.

2 Probabilistic and Possibilistic Networks

The basic idea underlying probabilistic as well as possibilistic networks is that under certain conditions a multi-dimensional distribution can be decomposed

without much loss of information into a set of (overlapping) lower-dimensional distributions. This set of lower-dimensional distributions is usually represented as a hypergraph, in which there is a node for each attribute and a hyperedge for each distribution of the decomposition. To each node and to each hyperedge a projection of the multi-dimensional distribution (a *marginal distribution*) is assigned: to the node a projection to its attribute and to a hypergraph a projection to the set of attributes connected by it. Thus hyperedges represent direct influences the connected attributes have on each other, i.e. how constraints on the value of one attribute affect the probabilities or possibilities of the values of the other attributes in the hyperedge.

Reasoning in such a hypergraph is done by propagating evidence, i.e. observed constraints on the possible values of a subset of all attributes, along the hyperedges. This can be done with *local computations*, usually restricted to a single hyperedge, if certain axioms are fulfilled [28].

Probability theory allows for local computations that are especially simple. Evidence entered into a node is first *extended* to the hyperedge along which it is to be propagated by multiplying the joint probability distribution associated with the hyperedge with the quotients of the posterior and prior probability of the values of the node. Then it can be *projected* to any other node contained in the hyperedge by simply summing out the other attributes (computing the new marginal distribution). A similar scheme can be derived for networks with directed edges to which a conditional probability distribution is assigned.

Possibilistic networks can be based on an interpretation of a degree of possibility that rests on the context model [6, 17]. In this model possibility distributions are interpreted as information-compressed representations of (not necessarily nested) random sets, a degree of possibility as the one-point coverage of a random set [21]. This interpretation allows to construct possibilistic networks in analogy to probabilistic networks. Only the propagation functions have to be replaced, namely the product (for extension) by the minimum and the sum (for projection) by the maximum.

Both types of networks, probabilistic as well as possibilistic, can be induced automatically from data. An algorithm for this task consists always of two parts: an evaluation measure and a search method. The evaluation measure estimates the quality of a given decomposition (a given hypergraph) and the search method determines which decompositions (which hypergraphs) are inspected. Often the search is guided by the value of the evaluation measure, since it is usually the goal to maximize or to minimize its value. In the following two sections we describe two search methods and several evaluation measures.

3 Search Methods

There is a large variety of search methods usable for learning inference networks. In principle any general heuristic search method is applicable, like hill climbing, simulated annealing, genetic algorithms etc. But to keep things simple, since our

emphasis is on evaluation measures, we consider only two methods: optimum weight spanning tree construction and greedy parent selection.

The construction of an optimum weight spanning tree was suggested already in [3]. An evaluation measure (in [3]: mutual information) is computed on all possible edges (two-dimensional subspaces) and then the Kruskal algorithm is applied to determine a maximum or minimum weight spanning tree.

Greedy parent selection is used in the K2 algorithm described in [4]. To narrow the search space and to avoid cycles in the resulting hypergraph a topological order of the attributes is defined. A topological order of the nodes of a directed graph satisfies: If there is a directed edge from a node A to a node B, then A precedes B in the order. Fixing a topological order restricts the permissible graph structures, since the parents of an attribute can be selected only from the attributes preceding it in the order. A topological order can either be stated by a domain expert or derived automatically [30].

Parent attributes are selected using a greedy search. At first an evaluation measure (in [4]: the g-function) is calculated for the child attribute alone, or — more precisely — for the hyperedge consisting only of the child attribute. Then in turn each of the parent candidates (the attributes preceding the child in the topological order) is temporarily added to the hyperedge and the evaluation measure is computed. The parent candidate yielding the highest value of the evaluation measure is selected as a first parent and is permanently added to the hyperedge. In the third step all remaining candidates are added temporarily as a second parent and again the evaluation measure is computed for each of the resulting hyperedges. As before, the parent candidate yielding the highest value is permanently added to the hyperedge. The process stops, if either no more parent candidates are available, a given maximal number of parents is reached or none of the parent candidates, if added to the hyperedge, yields a value of the evaluation measure exceeding the best value of the preceding step. The resulting hypergraph contains for each attribute a (directed) hyperedge connecting it to its parents (provided parents where added).

4 Evaluation Measures

In this section we review some evaluation functions that can be used for learning inference networks from data. All of them estimate the quality of single hyperedges and are based on the empirical probability or possibility distributions found in the database: If N is the number of tuples in the database and N_i the number of tuples in which attribute A has value a_i, then $P(a_i) = \frac{N_i}{N}$.

4.1 Measures for Learning Probabilistic Networks

The basic idea of several evaluation measures used for learning probabilistic networks is to compare the joint distribution with the product of the marginal distributions. This seems to be reasonable, since the more these two distributions differ, the more dependent the attributes are on each other. Other approaches include Bayesian estimation and minimization of description length.

The χ^2-Measure The χ^2-measure directly implements the idea to compare the joint distribution and the product of the marginal distributions by computing their squared difference. For two attributes A and B it is defined as

$$\chi^2 = \sum_{i,j} N \frac{(P(a_i)P(b_j) - P(a_i, b_j))^2}{P(a_i)P(b_j)},$$

where N is the number of tuples in the database to learn from.

This version of the χ^2-measure is sufficient, if the optimum weight spanning tree method is used, since then only two-dimensional edges have to be evaluated. But for learning hypergraphs, e.g. with the greedy parent search method, we need an extension to more than two attributes. Such an extension can be obtained in two ways, the first of which is to define for m attributes $A^{(1)}, \ldots, A^{(m)}$

$$\chi_1^2 = \sum_{i_1, \ldots, i_m} N \frac{\left(\prod_{k=1}^{m} P(a_{i_k}^{(k)}) - P(a_{i_1}^{(1)}, \ldots, a_{i_m}^{(m)}) \right)^2}{\prod_{k=1}^{m} P(a_{i_k}^{(k)})},$$

i.e. to compare the joint probability with the product of the single attribute marginal probabilities. The second extension is especially suited for learning directed hyperedges and consists simply in viewing the (candidate) parent attributes as one pseudo-attribute, i.e. if $A^{(1)}, \ldots, A^{(m-1)}$ are the (candidate) parents of attribute $A^{(m)}$, then

$$\chi_2^2 = \sum_{i_1, \ldots, i_m} N \frac{\left(P(a_{i_1}^{(1)}, \ldots, a_{i_{m-1}}^{(m-1)}) P(a_{i_m}^{(m)}) - P(a_{i_1}^{(1)}, \ldots, a_{i_m}^{(m)}) \right)^2}{P(a_{i_1}^{(1)}, \ldots, a_{i_{m-1}}^{(m-1)}) P(a_{i_m}^{(m)})}.$$

If not stated otherwise, all measures described for two attributes in the following can be extended in these two ways.

Entropy-based Measures In [3] the (two-dimensional) edges of a tree-decomposition of a multi-dimensional distribution are selected with the aid of *mutual information*. Under the name of *information gain* this measure was later used for the induction of decision trees [23, 24], which is closely related to learning inference networks (with directed edges): A hyperedge consisting of an attribute and its parents can be seen as a decision tree with the restriction that all leaves have to lie on the same level and all decisions in the same level of the tree have to be made on the same attribute.

Mutual information implements the idea to compare the joint distribution and the product of the marginal distributions by computing the logarithm of their quotient. For two attributes A and B mutual information is defined as

$$I_{\mathrm{mutual}} = \sum_{i,j} P(a_i, b_j) \log_2 \frac{P(a_i, b_j)}{P(a_i)P(b_j)} = H_A + H_B - H_{AB} = I_{\mathrm{gain}},$$

where H is the Shannon entropy [29]. It can be shown, that mutual information is always greater or equal to zero, and equal to zero, if and only if the joint distribution and the product of the marginal distributions coincide [18]. Hence it can be seen as measuring the difference between the two distributions. In the interpretation as information gain, it measures the information (in bits) gained about the value of one attribute from the knowledge of the value of the other attribute.

When using information gain for decision tree induction, it was discovered that information gain is biased towards many-valued attributes. To adjust for this bias the *information gain ratio* was introduced, which is defined as the information gain divided by the entropy of the split attribute [23, 24]:

$$I_{\text{gr}} = \frac{I_{\text{gain}}}{H_A} = \frac{I_{\text{gain}}}{-\sum_i P(a_i) \log_2 P(a_i)}.$$

Transfered to learning inference networks this means to divide the information gain by the entropy of the parent attributes. (Obviously this is only applicable when directed edges are used. Otherwise there would be no "split attribute" in contrast to the "class attribute.") In the two extensions to more than two attributes either the sum of the entropies of the marginal distributions of the parent attributes (first version) or the entropy of the marginal distribution of the pseudo-attribute formed from all the parent attributes (second version) forms the denominator.

An alternative is the *symmetric information gain ratio* defined in [20], which is the information gain divided by the entropy of the joint distribution:

$$I_{\text{sgr}}^{(1)} = \frac{I_{\text{gain}}}{H_{AB}} = \frac{I_{\text{gain}}}{-\sum_{i,j} P(a_i, b_j) \log_2 P(a_i, b_j)}.$$

Because of its symmetry this measure is also applicable for undirected edges. Another symmetric version that suggests itself is to divide by the entropy sum of the single attribute distributions, i.e.

$$I_{\text{sgr}}^{(2)} = \frac{I_{\text{gain}}}{H_A + H_B} = \frac{I_{\text{gain}}}{-\sum_i P(a_i) \log_2 P(a_i) - \sum_j P(b_j) \log_2 P(b_j)}.$$

It is easy to see that this measure leads to the same edge selections as the previous one. Nevertheless it is useful to consider both measures, since their effects can differ, if weighting is used (see section 5).

The measures discussed above are all based on Shannon entropy, which can be seen as a special case (for $\beta \to 1$) of *generalized entropy* [5]:

$$H^\beta(p_1, \ldots, p_r) = \sum_{i=1}^r p_i \frac{2^{\beta-1}}{2^{\beta-1} - 1} (1 - p_i^{\beta-1})$$

Setting $\beta = 2$ yields the *quadratic entropy*

$$H^2(p_1, \ldots, p_r) = \sum_{i=1}^r 2p_i(1 - p_i) = 2 - 2\sum_{i=1}^r p_i^2.$$

Using it in a similar way as Shannon entropy leads to the so-called Gini index:

$$\text{Gini} = \frac{1}{2}(H_A^2 - H_{A|B}^2) = \sum_{j=1}^{n_B} P(b_j) \sum_{i=1}^{n_A} P(a_i|b_j)^2 - \sum_{i=1}^{n_A} P(a_i)^2,$$

a well known measure for decision tree induction [2, 31]. Here only the second type of extension to more than two attributes is applicable. A symmetric ratio can be derived, but only for two attributes:

$$\text{Gini}_{\text{norm}} = \frac{H_A^2 - H_{A|B}^2 + H_B^2 - H_{B|A}^2}{H_A^2 + H_B^2}.$$

Bayesian Measures In the K2 algorithm [4] as an evaluation measure the g-function is used, which is defined as

$$g(A, \text{par}_A) = c \cdot \prod_{j=1}^{n_{\text{par}_A}} \frac{(n_A - 1)!}{(N_{.j} + n_A - 1)!} \prod_{i=1}^{n_A} N_{ij}!,$$

where A is an attribute and par_A the set of its parents. n_{par_A} is the number of distinct instantiations (value vectors) of the parent attributes that occur in the database to learn from and n_A the number of values of attribute A. N_{ij} is the number of cases (tuples) in the database in which attribute A has the ith value *and* the parent attributes are instantiated with the jth value vector, $N_{.j}$ the number of cases in which the parent attributes are instantiated with the jth value vector, that is $N_{.j} = \sum_{i=1}^{n_A} N_{ij}$. c is a constant prior probability, which can be set to 1, since usually only the relation between the values of the evaluation measure for different sets of parent attributes matters.

The g-function estimates (for a certain value of c) the probability of finding the joint distribution of the attribute and its parents that is present in the database. That is, assuming that all network structures are equally likely, and that, given a certain structure, all conditional probability distributions compatible with the structure are equally likely, it uses Bayesian reasoning to compute the probability of the network structure given the database from the probability of the database given the network structure.

MDL-based Measures Information gain can also be seen as measuring the reduction in the description length of a dataset, if the values of a set of attributes are encoded together (one symbol per tuple) instead of separately (one symbol per value). The *minimum description length principle* [26] in addition takes into account the information needed to transmit the coding scheme, thus adding a "penalty" for making the model more complex by enlarging a hyperedge. We consider the two types of minimum description length functions stated in [15] for decision tree induction.

Coding based on relative frequencies:

$$
\begin{aligned}
L_{\text{gain}}^{(1)} = \quad & \log_2 \frac{(N_{..} + n_A - 1)!}{N_{..}!(n_A - 1)!} + N_{..} H_A && L_{\text{prior}}^{(1)} \\
& - \sum_{j=1}^{n_B} \left(\log_2 \frac{(N_{.j} + n_A - 1)!}{N_{.j}!(n_A - 1)!} + N_{.j} H_{A|b_j} \right) && L_{\text{post}}^{(1)}
\end{aligned}
$$

The first term in each line states the costs for transmitting the frequency distribution. Intuitively, this is done by transmitting the page number for a code book listing all possible distributions of N cases on n_A attribute values. The second term in each line describes the costs to actually transmit the value assignments.

Coding based on absolute frequencies:

$$
\begin{aligned}
L_{\text{gain}}^{(2)} = \quad & \log_2 \frac{(N_{..} + n_A - 1)!}{N_{..}!(n_A - 1)!} + \log_2 \frac{N_{..}!}{N_{1.}! \cdots N_{n_A.}!} && L_{\text{prior}}^{(2)} \\
& - \sum_{j=1}^{n_B} \left(\log_2 \frac{(N_{.j} + n_A - 1)!}{N_{.j}!(n_A - 1)!} + \log_2 \frac{N_{.j}!}{N_{1j}! \cdots N_{n_A j}!} \right) && L_{\text{post}}^{(2)}
\end{aligned}
$$

Again the first term in each line describes the costs for transmitting the frequency distribution, the second term the costs for transmitting the value assignments. In this version the value assignments are also coded as a page number for a code book listing all possible assignments of values to cases for a given absolute frequency distribution. This measure is closely connected to the g-function described above. More precisely, it is $\log_2(g) = \log_2(c) - L_{\text{post}}^{(2)}$.

4.2 Measures for Learning Possibilistic Networks

In analogy to the probabilistic setting the idea of some of the measures for learning possibilistic networks is to compare the joint distribution with the minimum (instead of the product) of the marginal distributions. Other approaches are based on nonspecificity measures.

All measures described in this section are extended to more than two attributes in the first possible way and not by combining some of them into a pseudo-attribute. The reason is that in possibilistic networks edges are undirected, since it is difficult to define a conditional possibility distribution.

Comparison-based Measures For probabilistic networks both the χ^2-measure and mutual information compare directly the joint distribution and the product of the marginal distributions; the former by the difference, the latter by the quotient. Hence the idea suggests itself to apply the same scheme to possibilistic networks, replacing the product by the minimum and the sum by the maximum.

We thus obtain for two attributes A and B

$$
d_{\chi^2} = \sum_{i,j} \frac{(\min(\pi(a_i), \pi(b_j)) - \pi(a_i, b_j))^2}{\min(\pi(a_i), \pi(b_j))}
$$

as the analogon of the χ^2-measure, and

$$d_{\text{mi}} = -\sum_{i,j} \pi(a_i, b_j) \log_2 \frac{\pi(a_i, b_j)}{\min(\pi(a_i), \pi(b_j))}$$

as the analogon of mutual information. Since both measures are always greater or equal to zero, and zero if and only if the two distributions coincide, they can be seen as measuring the difference between the two distributions. Just as for information gain it may be a good idea to divide d_{mi} by the sum of the logarithms of the possibility degrees of the joint distribution to remove (or at least to reduce) a possible bias.

Nonspecificity-based Measures A possibilistic evaluation measure can also be derived from the U-uncertainty measure of *nonspecificity* of a possibility distribution [14], which is defined as

$$\text{nsp}(\pi) = \int_0^{\sup(\pi)} \log_2 |[\pi]_\alpha| d\alpha$$

and can be justified as a generalization of Hartley information [10] to the possibilistic setting [13]. $\text{nsp}(\pi)$ reflects the expected amount of information (measured in bits) that has to be added in order to identify the actual value within the set $[\pi]_\alpha$ of alternatives, assuming a uniform distribution on the set $[0, \sup(\pi)]$ of possibilistic confidence levels α [9].

The role nonspecificity plays in possibility theory is similar to that of Shannon entropy in probability theory. Thus the idea suggests itself to construct an evaluation measure from nonspecificity in the same way as information gain and (symmetric) information gain ratio are constructed from Shannon entropy.

By analogy to information gain we define *specificity gain* as

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

This measure is equivalent to the one defined in [9]. In addition, just like information gain ratio and symmetric information gain ratio, *specificity gain ratio*

$$S_{\text{gr}} = \frac{S_{\text{gain}}}{\text{nsp}(\pi_A)} = \frac{\text{nsp}(\pi_A) + \text{nsp}(\pi_B) - \text{nsp}(\pi_{AB})}{\text{nsp}(\pi_A)}$$

and *symmetric specificity gain ratio* in either of the two forms

$$S_{\text{sgr}}^{(1)} = \frac{S_{\text{gain}}}{\text{nsp}(\pi_{AB})} = \frac{\text{nsp}(\pi_A) + \text{nsp}(\pi_B) - \text{nsp}(\pi_{AB})}{\text{nsp}(\pi_{AB})}$$

or

$$S_{\text{sgr}}^{(2)} = \frac{S_{\text{gain}}}{\text{nsp}(\pi_A) + \text{nsp}(\pi_B)} = \frac{\text{nsp}(\pi_A) + \text{nsp}(\pi_B) - \text{nsp}(\pi_{AB})}{\text{nsp}(\pi_A) + \text{nsp}(\pi_B)}$$

can be defined.

5   Missing Values

In real world databases often a substantial number of values is missing. This, of course, poses problems for any learning algorithm for probabilistic networks, because it is not completely clear how to handle missing values when evaluating edges. Thus, when constructing a learning algorithm, it is often required that there are no missing values. In [4] and [12] this constraint is stated explicitly. But it is obvious that such an assumption does not lead to much, because it severely restricts the domain of application of the algorithm. Missing values are just too frequent in the real world for such a requirement.

When trying to handle missing values, the idea that comes to mind first is to ignore tuples possessing one or more of them, thus enforcing the above assumption. In some cases, where the number of tuples with missing values is small, this may be sufficient, but often a substantial part of the database has to be discarded in this way. Other approaches include replacing a missing value by a new distinct element *unknown* added to the domain of the corresponding attribute, thus transforming it into a normal value, or imputing the most frequent, an average, or a random value. But such approaches can distort the frequency distribution present in the database and hence may either lead to spurious dependences between attributes or conceal existing dependences.

Therefore in our implementation we refrained from using either of the above approaches, but tried the following scheme: Since the evaluation measures we use are local measures, they require only part of the tuple to be known in order to be computable. Hence, when evaluating a hyperedge, we ignore only those tuples in the database, in which a value is missing in one of the attributes contained in the hyperedge. Other attributes may be known or unknown, we do not care. In this way at least part of the information contained in a tuple with missing values can be used.

If an edge is evaluated using the described scheme, the resulting value of the measure for different edges can refer to different numbers of tuples. Hence for some measures, e.g. the g-function, the χ^2-measure and the reduction of description length measures, it is necessary to normalize their value, such that it refers to single tuples. To achieve this, the measure is simply divided by the number of tuples used to compute it.

In addition one may consider weighting the worth of an edge with the fraction of the tuples it was calculated on. This idea stems from learning decision trees with an information gain measure [24], where based on such a weighting a possible split attribute yielding a smaller gain is sometimes preferred to a split attribute yielding a higher gain, if due to a low probability of knowing the latter attribute the *expected gain* is higher for the first. By similar reasoning such weighting can be justified for learning inference networks. An edge may represent a stronger dependence of the connected attributes, but due to missing values it is less likely that this dependence can be exploited for inferences. In such a situation it may be preferable to use an edge connecting attributes whose dependence is weaker, but can be exploited more often. Nevertheless, in our experiments we

used unweighted measures, since weighting is not applicable to all measures and thus would have rendered some of the results incomparable.

For learning possibilistic networks, of course, missing values are no problem at all, since possibility theory was designed especially to handle such kind of uncertainty information. In the random set view, a missing value simply represents a random set containing all values of the underlying domain. Hence neither removing tuples nor additional calculations are necessary.

6   Evaluating Learned Networks

Learning probabilistic and possibilistic networks with local evaluation measures and search methods like greedy parent selection are heuristic methods. Hence it is not guaranteed that an optimal solution will be found. Thus the question arises: Can we find a way to assess a learned network, or at least a way to compare the quality of two networks?

For probabilistic networks a simple evaluation scheme can be derived from the idea underlying the Bayesian measure of the g-function described above. From a given network — dependence structure and (conditional) probabilities — the probability of each tuple in the database can be calculated. Multiplying these probabilities yields the probability of the database given the network structure, provided that the tuples are independent. If we assume all networks to have the same prior probability, the probability of the database given the network can be interpreted as a direct indication of the network quality. Although it is not an absolute measure, since we cannot determine an upper bound for this probability, networks can be compared with it.

The only problem with this method is the treatment of missing values, since for tuples with missing values no definite probability can be calculated. For our prototype program we decided on the following scheme: Every missing value of a tuple is instantiated in turn with each possible value, and for each resulting (completely known) tuple the probability is determined. Then the minimum, average, and maximum of these probabilities are computed. We thus arrive at a minimum, average, and maximum value for the probability of a database, of which the average may be the best to use. It is obvious that this method is applicable only, if the number of missing values per tuple is fairly small, since otherwise the number of tuples to be examined gets too large.

In theory such a global evaluation method can be used directly to learn a network. We only need to add a search method to traverse the space of possible solutions. Each candidate selected by the search method is then evaluated using the global evaluation function. But this is not very practical. The main reason is that evaluating a network in the way described can take fairly long, especially if there are missing values. If they abound, even a single network cannot be evaluated in reasonable time. Since during a search a large number of networks has to be inspected, the learning time can easily exceed reasonable limits. Nevertheless it may be worthwhile to examine such an approach.
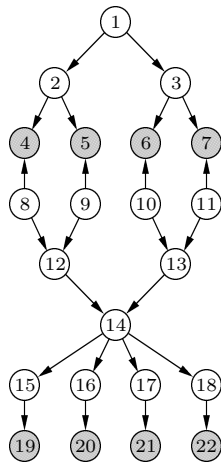
1 – parental error	12 – offspring ph.gr. 1
2 – dam correct?	13 – offspring ph.gr. 2
3 – sire correct?	14 – offspring genotype
4 – stated dam ph.gr. 1	15 – factor 40
5 – stated dam ph.gr. 2	16 – factor 41
6 – stated sire ph.gr. 1	17 – factor 42
7 – stated sire ph.gr. 2	18 – factor 43
8 – true dam ph.gr. 1	19 – lysis 40
9 – true dam ph.gr. 2	20 – lysis 41
10 – true sire ph.gr. 1	21 – lysis 42
11 – true sire ph.gr. 2	22 – lysis 43

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

Fig. 1. Domain expert designed network for the Danish Jersey cattle blood type determination example

We now turn to evaluating possibilistic networks. Unfortunately we cannot compute a degree of possibility for the whole database, but we can use a similar approach. From the propagation method of possibilistic networks it is obvious that the degree of possibility derivable from the network can only be greater or equal to the (true) degree of possibility derivable from the database. Hence, the better a network approximates the possibility distribution represented by the database, the smaller the sum of the possibility degrees over the tuples in the database should get. For tuples with missing values we use a similar approach as above. For each completely known tuple compatible with the tuple missing some values, the degree of possibility is determined from the network and the minimum, average, and maximum of these degrees is computed. Then these are summed for all tuples in the database. To be in accordance with the ideas underlying possibility theory, the maximum value may be the proper quality measure. If one commits to using the maximum, computation is significantly simplified, since a completely known tuple compatible with a tuple with missing values and having the maximum degree of possibility of all such tuples can easily be determined without inspecting all compatible tuples. Hence, with this restriction, a learning algorithm for possibilistic networks based on this global evaluation method may be a noteworthy alternative.

7 Experimental Results

The experiments described in this section were conducted with a prototype program called INES (Induction of NEtwork Structures). It contains the two search methods (optimum weight spanning tree construction and greedy parent selection) and all evaluation measures described above.

network	cond.	$\log_2(P_{\mathrm{avg}})$
indep.	0	-11632
orig.	24	-7451
db. prob.	24	-5105
χ^2	21	-5221
I_{gain}	21	-5221
$I_{\mathrm{sgr}}^{(1)}$	21	-5254
$\mathrm{Gini}_{\mathrm{norm}}$	21	-5505

network	cond.	$\log_2(P_{\mathrm{avg}})$
χ^2/N	33	-4373
I_{gain}	32	-4357
I_{gr}	20	-5243
$I_{\mathrm{sgr}}^{(1)}$	24	-4736
Gini	32	-4357
$\log_2(g)/N$	24	-4620
$L_{\mathrm{gain}}^{(1)}/N$	24	-4704
$L_{\mathrm{gain}}^{(2)}/N$	26	-4620

Table 1. Evaluation of probabilistic networks obtained by optimum weight spanning tree construction (left bottom) and by greedy parent selection (right) on the Danish Jersey cattle data. As a reference point evaluations of a network with independent nodes, of the original expert designed network, and of the expert designed network with probabilities determined from the database (left top) are added.

Although we tested INES on several databases from the UCI machine learning repository (e.g. flags, solar flare, mushroom, vote etc.), we chose to present here the results obtained on the Danish Jersey cattle blood type determination example [25], because it has the advantage that there is a probabilistic network designed by domain experts (see figure 1), which can be used as a baseline for result evaluation. Results on other datasets do not differ significantly.

The Danish Jersey cattle blood type determination example consists of the domain expert designed network, whose structure is shown in figure 1, and a database containing 500 tuples over the twenty-two attributes of the network. Only eight of the attributes, those shaded in the network, are actually observable. Several tuples of the database contain missing values.

As a baseline for comparisons we first evaluated a network without any edges (isolated nodes), the domain expert designed network, and the domain expert designed network with probabilities determined from the database. Their evaluation shows that the database seems to be a little distorted and not really fitting the domain expert designed model, since the evaluation of the original network is considerably worse than that of the network with adjusted probabilities.

We then tested inducing probabilistic networks on this dataset. For each of the symmetric measures described in section 4 (I_{gain}, I_{sgr}, $\mathrm{Gini}_{\mathrm{norm}}$, and χ^2), we constructed an optimum weight spanning tree and evaluated it on the database. The results are shown in the bottom left of table 1 (since we did not use weighting, the results for the two symmetric information gain ratios are the same, hence only one is shown). Although they are restricted to 21 edges because of the tree structure, they come fairly close to the evaluation of the adapted original network. Inspecting the learned networks in more detail reveals that their structure is indeed very close to the domain expert designed network.

In a third step we induced networks with the greedy parent selection method. We selected a topological order compatible with the domain expert designed net-

network	cond.	$\sum \pi_{\min}$	$\sum \pi_{\mathrm{avg}}$	$\sum \pi_{\max}$
indep.	0	139.8	141.1	158.2
db. poss.	24	137.3	137.7	157.2
d_{χ^2}	21	120.3	122.5	143.5
d_{mi}	21	117.6	119.4	144.3
S_{gain}	21	123.3	124.9	148.8
$S_{\mathrm{sgr}}^{(1)}$	21	121.1	123.8	148.3
d_{χ^2}	35	122.3	123.2	145.2
d_{mi}	33	115.6	118.1	147.2
S_{gain}	35	122.9	123.8	146.1
S_{gr}	27	129.9	131.4	154.1
$S_{\mathrm{sgr}}^{(1)}$	33	123.6	124.7	147.2

Table 2. Evaluation of possibilistic networks obtained by optimum weight spanning tree construction (top) and by greedy parent selection (bottom) on the Danish Jersey cattle blood type determination data.

work and restricted the number of parents to two. To compute the evaluation measures for attributes with two parents, we combined the parents into one pseudo-attribute, i.e. we extended the measure in the second possible way. Evaluations of the learned networks are shown on the right in table 1. With the exception of the information gain ratio network they all perform better than the original structure. A closer inspection reveals that they do so by exploiting additional dependences present in the database but not in the domain expert designed network.

From this table one may infer that information gain, Gini index and χ^2-measure lead to the best results, since given the networks learned with these measures, the average probability of the database is highest. But taking into account the number of conditions selected, which is highest for these measures, the suspicion arises that the good evaluation results are obtained by "over fitting" the data. This hypothesis was confirmed by an experiment on two artificial datasets generated from the domain expert designed network. On the test dataset the evaluation results of the networks learned with these measures where considerably lower than on the dataset they were learned from. The effect seems to be less pointed for information gain than for Gini index and χ^2-measure. Nevertheless, the bias in favour of many-valued attributes was clearly visible, since with information gain the *offspring genotype* attribute (6 values) was selected as a parent attribute for the *lysis* attributes instead of the *factor* attributes (2 values) as in the domain expert designed network. The results also confirmed that forming some kind of ratio reduces the bias.

Evaluations of learned possibilistic networks are shown in table 2. It is not surprising that the expert designed network (with possibility degrees determined from the database) performs badly, since the possibilistic scheme exploits a dif-

ferent type of dependence. But it is remarkable that allowing larger edges to be learned by using the greedy parent selection method seems not to improve the results over optimum weight spanning trees, although this may be due to the restrictions imposed by the topological order. The strength and weaknesses of the measures seem to be similar to those of the analogous measures for learning probabilistic networks.

8   Conclusions

In this paper we considered two search methods (optimum weight spanning tree construction and greedy parent selection) and a large number of local evaluation measures for learning probabilistic and possibilistic networks. The experimental results, which we obtained with the prototype program INES, show that a problem of some evaluation measures is that they lead to the selection of too large edges (in terms of the number of attributes as well as in terms of the number of attribute values), resulting in some kind of "over fitting" (information gain, Gini index, χ^2-measure). For probabilistic networks the best results seem to be achievable with the symmetric information gain ratio, the g-function and the minimum description length measures. For possibilistic networks d_{mi}, the analogon of mutual information, seems to yield the best results.

Acknowledgments

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. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA, 1984
3. C.K. Chow and C.N. Liu. Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Trans. on Information Theory* 14(3):462–467, IEEE 1968
4. G.F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning* 9:309–347, Kluwer 1992
5. Z. Daróczy. Generalized Information Functions. *Information and Control* 16:36–51, 1970
6. 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.
7. 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

8. J. Gebhardt and R. Kruse. Learning Possibilistic Networks from Data. *Proc. 5th Int. Workshop on AI and Statistics*, 233–244, Fort Lauderdale, 1995

9. 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

10. R.V.L. Hartley. Transmission of Information. *The Bell Systems Technical Journal* 7:535–563, 1928

11. D. Heckerman. *Probabilistic Similarity Networks*. MIT Press 1991

12. 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

13. M. Higashi and G.J. Klir. Measures of Uncertainty and Information based on Possibility Distributions. *Int. Journal of General Systems* 9:43–58, 1982

14. G.J. Klir and M. Mariano. On the Uniqueness of a Possibility Measure of Uncertainty and Information. *Fuzzy Sets and Systems* 24:141–160, 1987

15. I. Kononenko. On Biases in Estimating Multi-Valued Attributes. *Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining*, 1034–1040, Montreal, 1995

16. R. Kruse, E. Schwecke, and J. Heinsohn. *Uncertainty and Vagueness in Knowledge-based Systems: Numerical Methods.* Springer, Berlin 1991

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

18. S. Kullback and R.A. Leibler. On Information and Sufficiency. *Ann. Math. Statistics* 22:79–86, 1951

19. 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

20. R.L. de Mantaras. A Distance-based Attribute Selection Measure for Decision Tree Induction. *Machine Learning* 6:81–92, Kluwer 1991

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

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

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

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

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

26. J. Rissanen. A Universal Prior for Integers and Estimation by Minimum Description Length. *Annals of Statistics* 11:416–431, 1983

27. 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

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

29. C.E. Shannon. The Mathematical Theory of Communication. *The Bell Systems Technical Journal* 27:379–423, 1948

30. M. Singh and M. Valtorta. An Algorithm for the Construction of Bayesian Network Structures from Data. *Proc. 9th Conf. on Uncertainty in AI*, 259–265, Morgan Kaufman, 1993

31. L. Wehenkel. On Uncertainty Measures Used for Decision Tree Induction. *Proc. IPMU*, 1996