Constructing Belief Networks From Realistic Data

Yun Peng*
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250

Zonglin Zhou
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, 100080 China

Sungzoon Cho*
Department of Computer Science and Engineering
Pohan University of Science and Technology
San 31 Hyojadon, Pohan, 790-784, Korea

[1] **Abstract**

In this paper, we report our investigation on the behavior of some Bayesian belief network learning methods when applied to realistic case data which do not accurately reflect the probability distribution of the underlying domain. The investigation focuses on two types of such data of practical interest, namely data excluding null or normal cases and data contaminated by noise, and their effects on the performance of two learning methods, the K2 algorithm, a representative method of Bayesian approach and the extended Hebbian learning (EHL) method, representing those of neural network approach. Both analytical and experimental results show that when null cases are removed from the case database, the EHL method is able to accurately construct the underlying causal structure whereas K2 algorithm may fail to properly handle root nodes. On the other hand, the EHL method is noise sensitive while K2 has certain inherent noise-resistance capability. Suggestions to extend both K2 and EHL to better cope with these problems are also made, and their effectiveness tested through a systematic computer simulation experiment.

[1]* This work was supported in part by NSF Grant IRI-9309136 to Yun Peng, and POSTECH PIRL Grant B94918 and KOSEF Grant 941-0900-031-02 to Sungzoon Cho.

1 Introduction

Active research work has been conducted in recent years to develop effective methods to construct Bayesian belief networks from case data. Examples of recent development include methods which infer a causal network structure that best fits a given database of cases in terms of conditional independence between variables (e.g., IC-algorithm of Pearl and Verma[11], PC-algorithm of Spirtes, Glymour and Scheines[16]); Bayesian methods which construct a belief network with the most probable structure, given a database of cases (e.g., K2 algorithm of Cooper and Herskovits[3], and the work of Heckerman, Geiger and Chickering that combines prior knowledge and case data in Bayesian learning[5]); and methods using minimum description length (MDL), a principle that seeks a balance between the accuracy and parsimony of the learning results[7,17]. All these methods employ some form of heuristic search to obtain the (sub)optimal results. Taking advantage of the structural similarity between neural networks and belief networks and the existence of a large repertoire of neural network learning techniques, other researchers have adopted some neural network learning methods to belief networks. This includes Neal's method based on Boltzmann machine model[8], and our own learning method which is an extension of the well-known Hebbian rule[13,14].

Most studies on learning belief networks from case data in the past have been concentrated on developing methods that can construct belief networks that best fit the given case databases according to some measurement. Therefore, any of a number of well-developed learning methods can be used to accurately construct the causal structure of a belief network, provided the case data used in learning are sufficiently accurate in reflecting the probability distribution of the domain. In fact, to empirically establish the validity of their respective methods, researchers often use databases of ideal case data randomly generated from a *target* belief network as learning inputs and then compare the learned network with the target network. If the size of such a database is sufficiently large, then the database reflects accurately the probability distribution of the underlying target networks.

Unfortunately, for various reasons, case data taken from real-world applications are rarely pure or accurate, and little attention has been paid to the behavior of these learning methods when applied to inaccurate case data. For example, an ideal case description in a case database is often a complete instantiation of all variables involved. However, in many applications not every variable is observed or reported in every case. Therefore, some cases may contain uninstantiated variables. This is the so-called *missing value* problem. Moreover, some variables may never be observed or included while collecting case data, but postulation of their existence may help to improve a belief network's fitting of the given observed data. Variables of this nature are often referred to as *hidden variables*. Problems of missing values and hidden variables are closely related because, as suggested by Cooper and Herskovits[3], hidden variables can be represented by a node in a belief network with missing value for every case in the database. Learning from case data with missing values and hidden variables has received some attention from researchers[3]. We also note that the Boltzmann machine model of Neal's seems to be an ideal vehicle for handling hidden variables because this model allows hidden nodes (i.e., nodes which cannot be instantiated externally) as an integrated part of the network[8].

In this paper we focus on two other types of realistic but not ideal case data, which are of important practical interest but have not received rigorous investigation. They are data in which all null cases are excluded and data contaminated by noise, referred to as non-null data and noisy data, respectively in this paper. Data contamination can be caused, among other things, by human and instrumental errors at every step of data collection and processing. Properly handling contaminated or noisy data constitutes a common challenge to all learning tasks. Null cases can be better understood in the setting of abductive applications such as medical diagnosis. In such applications, a node in a belief network is often treated as a binary variable, representing whether a diagnostic entity (a disorder, a symptom, or an intermediate state) is normal (value 0) or abnormal (value 1). Each case can thus be represented as a vector whose elements are the instantiations of these binary variables. In particular, a null case is a $\mathbf{0}$ vector, representing a case in which all variables are normal. Case data taken from a doctor's office usually contain very few, if any, null or normal cases because people usually do not see doctors if they do not feel ill. In other words, databases in these applications are collections of abnormal, i.e., non-null cases.

The question then is: Can the existing learning methods developed for ideal case data be used to construct causal structures from non-null data and noisy data? To answer this question, in this paper the learning behaviors of some existing representative methods are systematically investigated, both analytically and experimentally. In the investigation, we identify the main obstacles these data impose on the learning tasks, and make suggestions to extend these methods to overcome these obstacles.

Two existing learning methods are selected for this study. They are the K2 algorithm of Cooper and Herskovits[3], a representative of methods taking the Bayesian approach, and our own Extended Hebbian Learning (EHL) method, representing methods taking the neural network approach[13,14]. This work is restricted to a special class of belief networks, known as noisy-or belief networks, and to only binary variables, because, among other reasons such as practical usefulness, neural network based methods such as the EHL and Neal's Boltzmann machine model only work with such belief networks.

The rest of this paper is organized as follows. Section 2 provides background information, including a brief description of the formulation of Bayesian belief networks in general and noisy-or belief networks in particular, and summaries of the two selected learning methods. Sections 3 and 4 analyze the learning methods' behaviors and suggest possible modifications for handling non-null data and noisy data, respectively. To systematically test the effectiveness of suggested modifications, a comprehensive computer experiment is conducted and its results presented in Section 5. Finally, Section 6 concludes with suggestions for future research.

2 Background

In this section we provide a brief descriptions of general Bayesian belief networks and a special class of belief networks known as noisy-or belief networks, and summarize the learning rules for the K2 algorithm and the EHL method.

2.1 General and Noisy-Or Belief Networks

In the most general form, a Bayesian belief network of n variables consists of a directed acyclic graph (DAG) of n nodes and a number of arcs, referred to as the network structure, and a conditional probability distribution (CPD) associated with the graph[9,10]. Nodes in a DAG correspond to variables, denoted $v_i, i = 1, \cdots, n$. The upper case V_i is used to denote an instantiation of v_i, and $V = (V_1, \cdots, V_n)$ a vector representing a given case. In this work, we take the causal interpretation of the arcs and consider an arc $< v_j, v_i >$ as a *causal link* representing the direct cause-effect association from v_j to v_i. The CPD consists of $P(v_i|\pi_i)$, the conditional probability distribution for each node v_i, given its parent set π_i. When v_i does not have any parent, i.e., v_i is a root in the network, $P(v_i|\pi_i)$ becomes $P(v_i)$, the prior probability of v_i. A conditional independence assumption is made for belief networks: $P(v_i|\pi_i, S) = P(v_i|\pi_i)$, where S is any set of variables excluding v_i and its descendants. With this assumption, variable v_i not only is independent of any other variables which are not causally related to it, but also conditionally independent of all of its causal ancestors, given π_i. Also, one can derive from this assumption that all roots are independent of each other.

The noisy-or networks[9,10,12] are a special class of general belief networks. Instead of using the conditional probabilities $P(v_i|\pi_i)$, this type of network associates a single probability measure, called the *causal strength*, to each causal link to capture the degree of uncertainty of the causal relation. Besides reducing tremendously the number of probabilities required, the one-to-one correspondence between causal links and causal strengths makes noisy-or networks more natural to humans than those using $P(v_i|\pi_i)$. This is particularly valuable when human experts are involved in acquisition and assessment of causal knowledge for belief network constructions.

According to the formulation of noisy-or networks of binary variables by Peng and Reggia[12], the causal strength, denoted c_{ji} for link $< v_j, v_i >$, is defined as follows. Let $\kappa_{ji} = 1$ denote a special event, called *causation*, that "$v_j = 1$ actually and directly causes $v_i = 1$". Then,

$$c_{ji} = P(\kappa_{ji} = 1 | v_j = 1), \tag{1}$$

representing the strength of the causal association between v_j and v_i†. [2] When there does not exist a causal link from v_j to v_i, then $c_{ji} = 0$. In other words, v_j is a direct cause of v_i and thus there is a link from v_j to v_i in DAG if and only if $c_{ji} > 0$. Therefore, the network structure (i.e., the DAG) can be completely depicted by the causal strengths c_{ji} for all j, i.

The causal strengths c_{ji} are a special way to represent the *CPD* for simpler causal relations, and they can easily be converted to conditional probabilities as follows. For any non-root node v_i,

$$P(v_i = 1|\pi_i) = 1 - \prod_{v_j \in \pi_i} (1 - c_{ji} v_j). \tag{2}$$

An implication of this is that any method that can learn the structures of general belief networks from case data can also learn the structure of noisy-or networks.

Strictly speaking, learning belief networks includes learning both DAG, the network structure, and the probability distribution (*CPD* or c_{ij}). Learning the probability distribution is relatively easy and straightforward after the correct network structure is obtained. As can be seen shortly, the EHL method learns both DAG and c_{ji}'s at the same time, while the K2 method only learns DAG (a separate method is provided by Cooper and Herskovits to learn *CPD* after DAG is obtained[3]). For these reasons, this study will concentrate only on learning the network structures.

2.2 The Two Learning Methods

The K2 algorithm of Cooper and Herskovits can learn the structures of general belief networks of multi-valued variables from a given fixed case database. The EHL method we developed previously can only learn noisy-or networks of binary variables. Our experience shows that, when used to learn noisy-or networks, both methods exhibit very similar accuracy but EHL is substantially faster than the K2 method[14]. Both methods require a pre-existing order among all nodes in such a way that only those nodes which precede a particular node can possibly be chosen as its parents during learning. For clarity, in this paper we use the node subscripts to specify this node order. Therefore, node v_j can be chosen as a parent of node v_i only if $j < i$.

[2]† Others[9,10] attach the quantity "$1 - c_{ji}$" to the causal link from v_j to v_i, representing the probability that $v_j = 1$ is ineffective in forcing v_i to be 1.

The K2 Algorithm

The K2 algorithm aims at constructing the most probable network structure B_S, given a database of cases Γ, i.e., the one with the highest $P(B_S|\Gamma)$. Since K2 is only interested in the relative ranking of all possible structures, it uses the joint probability $P(B_S,\ \Gamma)$, not $P(B_S|\Gamma)$ to measure the quality of a given structure B_S. Let $r_i > 1$ be the number of values variable v_i can have and q_i the number of distinct instantiation patterns its parent set π_i has in Γ. Then, under a set of assumptions, it can be shown that[3]

$$P(B_S, \Gamma) = P(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + ri - 1)!} \prod_{k=1}^{r_i} N_{ijk}!, \tag{3}$$

where N_{ijk} is the number of cases in Γ in which v_i has the k^{th} of the r_i values, and π_i is the j^{th} of the q_i instantiations, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

K2 employs a greedy algorithm to search for a B_S which is (sub)optimal measured by Eq. (3). It constructs the parent set π_i for each $v_i, i = 2, 3, \cdots, n$ one at a time as follows. For a given v_i, it successively adds a new node $v_j, j < i$ into the current π_i, if adding v_j into π_i would cause the most increase of the posterior probability of the resulting structure among all unselected predecessors of v_i. This process of constructing π_i starts with $\pi_i = \emptyset$ and continues until no addition of any node into the current π_i can increase the probability. When all structures B_S are assumed to have equal prior probability, the following equation, a localized version of Eq. (3), is used to compute the probability at each step

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! = \prod_{j=1}^{q_i} \frac{N_{ij0}! \cdot N_{ij1}!}{(N_{ij} + 1)!}. \tag{4}$$

The second equality in Eq. (4) above is the formula for networks of binary variables ($r_i = 2$) where N_{ij0} and N_{ij1} are the numbers of cases in Γ in which π_i is the j^{th} of the q_i instantiations and $v_i = 0$, and 1, respectively. The algorithm K2 itself goes as follows.

for $i = 1$ *to* n **do**

$\pi_i \leftarrow \emptyset$

for $i = 2$ *to* n **do**

repeat

select $v_j \in \{v_1 \cdots v_{i-1}\} \backslash \pi_i$ that maximizes $g(i, \pi_i \cup \{v_j\})$

$\Delta \leftarrow g(i, \pi_i \cup \{v_j\}) - g(i, \pi_i)$

if $\Delta > 0$ **then** $\pi_i \leftarrow \pi_i \cup \{v_j\}$

until $\Delta \leq 0$ **or** $\pi_i = \{v_1 \cdots v_{i-1}\}$.

The EHL Method

As noted earlier, the structure of a noisy-or network can be completely depicted by the causal strengths c_{ji} for all $j < i$. Consequently, if all of the c_{ji}'s can be correctly learned, then the structure of the learned network can be obtained by drawing causal links only between those $< v_j, v_i >$ whose learned causal strengths are greater than zero (or a very small positive number chosen as the cut-off value). The extended Hebbian learning (EHL) method we developed in the past is aimed at learning these causal strengths[13,14]. This method is an adoption of the well-known Hebbian law of neural network learning to the framework of noisy-or belief networks. The learning process takes as inputs the case vectors and updates the parameters of the learning network incrementally with each input. Let $p_j = P(v_j = 1)$ denote the prior probabilities of v_j and c_{ji} the causal strength of $< v_j, v_i >$ in the underlying network that generates the case inputs. The learning network, also of n nodes, uses primarily two classes of learning parameters. They are

1) $x_j, j = 1, \cdots, n$, to be trained to p_j; and

2) $w_{ji}, j < i$, to be trained to $P(\kappa_{ji} = 1) = c_{ji} \cdot p_j$.

Then the learned causal strengths y_{ji} can be directly computed by w_{ji}/x_j at any time. When x_j and w_{ji} approach p_j and $c_{ji} \cdot p_j$, respectively, y_{ji} approaches c_{ji}. Correspondingly, EHL uses two learning rules for x_j and w_{ji}, respectively. With the case $V(t)$ presented to the learning network at time t,

Rule 1: $$x_j(t) = x_j(t-1) + \alpha(V_j(t) - x_j(t-1)),\ t > 0; \quad \text{and} \tag{5}$$

Rule 2: $$w_{ij}(t) = w_{ji}(t-1) + \alpha(V_j(t)V_i(t)q_{ji}(t) - w_{ji}(t-1)), \quad \text{where} \tag{6}$$

$$q_{ji}(t) = \frac{y_{ji}(t-1)}{1 - \prod_{k<i}(1 - y_{ki}(t-1)V_k(t))}, \tag{7}$$

and $0 < \alpha < 1$ is the learning rate.

It can easily be seen from Rule 2 that w_{ji} increases only if both $V_j(t)$ and $V_i(t)$ equal one in a given case input $V(t)$. This is in accordance to the spirit of the renowned Hebbian law which says that the synaptic weight between two neurons increases only when both neurons are activated[4,6]. However, Rule 2 differs from the standard Hebbian law by its inclusion of the factor q_{ji}. It can be shown that without this factor (i.e., q_{ji} is fixed to 1 in Eq. (6)), Rule 2 trains w_{ji} to the joint probability $P(v_j = 1, v_i = 1)$, not the probability of the causation $P(\kappa_{ji} = 1)$.

The factor q_{ji} not only guarantees w_{ji} be properly trained to $P(\kappa_{ji} = 1)$, it also introduces competition into learning. Note that when $V_i(t) = 1$, $w_{ki}(t)$ for all nodes $v_k, k < i$, with $V_k(t) = 1$ will increase. The amounts of increase depend on their respective $q_{ki}(t)$ which, as can be seen in Eq. (7), have the same denominator. These potential parents of v_i are thus grouped together to compete with each other in increasing their w_{ki} with the respective currently learned causal strengths $y_{ki}(t-1)$ as the competition factors, normalized by the common denominator. This competition-based update of Rule 2 is crucial for the effectiveness and efficiency of this method[14].

The learning algorithm of EHL is simply an iteration of the following three steps that update all learning variables with each case input $V(t)$:

update $q_{ji}, i = 2, \cdots, n, j < i$ by Eq. (7);

update $x_j, j = 1, \cdots, n$ by Eq. (5);

update $w_{ji}, i = 2, \cdots, n, j < i$ by Eq. (6).

Note here when using Eq. (7) in Step 1, $y_{ji}(t-1)$ are computed by $w_{ji}(t-1)/x_j(t-1)$. The iteration starts with an initial network in which x_i and $w_{ji}, i = 1, \cdots, n$ and $j < i$ are assigned with very small random positive values. In other words, there is a link from v_j to v_i for each pair $< v_j, v_i >, j < i$, in the initial network and all predecessors of node v_i are its potential parents when learning starts. The learning process stops when the network reaches a stable state. The final network is then obtained by retaining only those links whose learned causal strengths y_{ji} are greater than a given small cut-off value.

It can be shown that with a given fixed database of cases Γ, this method is able to construct a noisy-or belief network with its joint probability distribution of the n variables very close to that of Γ. Stated formally, let $\widetilde{V} = (\tilde{V}_1 \ldots, \tilde{V}_n)$ denote a state of the learning network. Then the two learning rules will maximize the function[14]‡ [3]

$$L = \ln \prod_{V^r \in \Gamma} P(\widetilde{V} = V^r) \tag{8}$$

which measures the likelihood that the learned network is to produce the case data in Γ. The maximization of Eq. (8) is achieved by moving x_j's and w_{ji}'s along the gradient ascent of L by applications of Rules 1 and 2 upon the presentation of each of the case $V \in \Gamma$. When the system settles down into a stable state that globally maximizes Eq. (8), $x_j \rightarrow p_j$ and $y_{ji} \rightarrow c_{ji}$, provided Γ truly reflects the probability distribution of the underlying target network[14].

2.3 Summary

K2 and EHL represent two vastly different approaches toward learning belief networks from case data. K2 is aimed at constructing an optimal network structure measured by the posterior probability while the optimization measure used by EHL is a likelihood function L. Learning by K2 is done in a sequential manner in which the causal links (parent-child relations) are determined one at a time. In contrast, EHL tries to construct an optimal network in a parallel manner with all links simultaneously updating their weights as the network moving along the gradient ascent of L. As a result, K2 only learns the network structure while EHL learns both the structure and the probability distribution at the same time. Nevertheless, both K2 and EHL are proven to be effective learning methods for noisy-or belief network structures with ideal case data. However, it is unclear *a priori* how well each of these two methods will perform when applied to case data from the real world which are not ideal.

[3]‡ This function is called a Liapunov function in the studies of dynamic systems, or an energy function, due to its analogy to energy in thermodynamics. This function is the same as the one used in Neal's Boltzmann machine model. However, the learning rules derived for EHL is different from that of Neal's in that i) Neal's model allows hidden nodes but EHL does not; and ii) the equations of motion of EHL (Eqs. (5), (6), (7)) has a competition factor q_{ji} but Neal's model does not.

In what follows, we will systematically investigate, both analytically and experimentally, the behavior of these two methods when case data are noisy and non-null. For notational clarity, "ˆ" and " *" are used to denote terms and variables associated with non-null and noisy data, respectively, in the rest of the paper. For instance, we will use Γ to denote the databases of ideal cases, and $\hat{\Gamma}$ and Γ^* the databases of non-null and noisy cases cases, respectively.

Two noisy-or networks of different structures, given in Figures 1 and 2 below, are used in the experiments to generate Γ's of pure cases. $\hat{\Gamma}$ or Γ^*, when needed, are obtained by removing null cases from Γ or adding noise into Γ. To accurately reflect the probability distributions of the target networks, databases used in the experiments all contain sufficiently large number of cases. These two networks are also used as the target networks to be compared with the DAG's learned by the two methods. The first target network is a 10 by 10 bipartite network where the 10 nodes at the top layer are the roots with prior probabilities ranged from 0.014 to 0.098. There are 35 causal links in this network with the causal strengths c_{ji} ranged from 0.11 to 0.96. The second network is substantially larger and more complex. It has 37 nodes (12 roots, 8 leaves and 17 intermediate nodes) and 46 links, and the structure of the "Alarm belief network" used in[3]. The prior probabilities of roots and the causal strengths of this network are generated randomly by us for the experiment, ranged from 0.024 to 0.111, and from 0.11 to 0.92, respectively.

Figure 1 is about here

Figure 2 is about here

---

3 Learning With Non-Null Case Data

In this section we investigate how the exclusion of null cases from the case database Γ affects the learning behaviors of the two methods. Let $V_\emptyset = \mathbf{0}$ denote the null case, $\hat{\Gamma} = \{V^r \neq V_\emptyset | V^r \in \Gamma\}$ the database in which all null cases are removed, and $p_\emptyset = P(V_\emptyset|\Gamma)$ the probability of null cases in Γ. It is obvious that the n variables have different probability distributions in Γ and $\hat{\Gamma}$. First, the prior probabilities of variables in $\hat{\Gamma}$ become greater than that of Γ. More importantly, the interdependencies between variables are also changed when going from Γ to $\hat{\Gamma}$. For example, consider two independent variables v_j and v_i in Γ, i.e., $P(v_j, v_i|\Gamma) = P(v_j|\Gamma)P(v_i|\Gamma)$. It can easily be seen that $P(v_j = 1, v_i = 1|\hat{\Gamma}) < P(v_j = 1|\hat{\Gamma})P(v_i = 1|\hat{\Gamma})$ because the total number of cases in Γ is greater than that of $\hat{\Gamma}$, but the numbers of cases in which $v_j = 1$, $v_i = 1$, and both $v_j = v_i = 1$ remain unchanged when going from Γ to $\hat{\Gamma}$. Therefore, v_j and v_i are not independent of each other in $\hat{\Gamma}$. An immediate implication of this is that root variables are no longer independent of each other when null cases are excluded from the database. As will be shown by the experiment below, this change of probability distribution may cause some serious problems for the K2 algorithm when $p_\emptyset$ is large, whereas the EHL method is apparently unaffected.

A computer simulation experiment was conducted to test the effect of $p_\emptyset$ on the learning results. For each of the two target networks, three different $p_\emptyset$ were used to generate the case data for the experiment. These $p_\emptyset$ came from different sets of prior probabilities of the roots. Each of the six databases generated contained 10,000 non-null cases. All the null cases randomly generated during the input preparation were discarded. The cut-off value for EHL in the experiment was set to 0.001. Table 1 below gives the experiment results from the two methods applied to the six non-null databases. Regardless of the value of $p_\emptyset$, EHL performed very well with only one or two extra links being added in each of the six experimental runs. However, this was not the case for K2. A significant number of extra links was mistakenly added except when $p_\emptyset$ was very small. Even when $p_\emptyset$ was as small as about 0.05, the number of extra links learned by K2 was not negligible. Moreover, almost all of these extra links were found to be between roots.

Table 1 is about here

The experiment suggests that the underlying causal structure can be constructed accurately by EHL using either Γ or $\hat{\Gamma}$. To formally establish this, we show that, when the two learning rules of EHL are applied to a null case, the learned causal strength y_{ij} remains the same.

Theorem 1: $\forall j < i, y_{ji}(t) = y_{ji}(t-1)$ when the t^{th} learning input $V(t)$ is a null case.

Proof: Since $V(t) = \mathbf{0}$, we have, by Rules 1 and 2 of EHL (Eqs. (5), (6))

$$x_j(t) = x_j(t-1) + \alpha(0 - x_j(t-1)) = (1-\alpha)x_j(t-1).$$

$$w_{ji}(t) = w_{ji} + \alpha(0 - w_{ji}(t-1)) = (1-\alpha)w_{ji}(t-1).$$

Then by the definition of y_{ji}, $y_{ji}(t) = w_{ji}(t)/x_j(t) = w_{ji}(t-1)/x_j(t-1) = y_{ji}(t-1)$. □

Theorem 1 holds regardless when and how many null cases are used during learning. Therefore, y_{ji} trained by Γ always equals $\hat{y}_{ji}$ trained by $\hat{\Gamma}$, and thus the structure found using $\hat{\Gamma}$ is the same as the one found using Γ, although $\hat{x}_j$ and $\hat{w}_{ji}$ are usually different from their counterparts x_j and w_{ji}.

Now we analyze the behavior of K2 when $\hat{\Gamma}$ is used. Consider two roots v_j and $v_i, j < i$. As roots, v_j and v_i are independent of each other in Γ. Recalling the algorithm of K2 given in Subsection 2.2, to select v_j as a parent of v_i, it is necessary that adding v_j into π_i increases the probability measured by function $g(.)$. We next show that K2 will never select v_j as a parent of v_i when using Γ, but may do so if $\hat{\Gamma}$ is used. This is done by first showing that $R = g(i, \{j\})/g(i, \emptyset) < 1$ with Γ, and then that $\hat{R} = \hat{g}(i, \{j\})/\hat{g}(i, \emptyset) > R$ with $\hat{\Gamma}$ in the next two theorems. Due to their lengths, the proofs of the two theorems are given in the appendix.

Theorem 2: If v_j and $v_i, j < i$, are independent of each other in Γ where $N = |\Gamma| \gg 1$, then

$$R = g(i, \{j\})/g(i, \emptyset) < 1. \tag{9}$$

Theorem 3: With the same assumptions made in Theorem 2,

$$\frac{\hat{R}}{R} = \frac{\hat{g}(i, \{j\})}{\hat{g}(i, \emptyset)} / \frac{g(i, \{j\})}{g(i, \emptyset)} > 1 \tag{10}$$

Combining Theorems 2 and 3, it is possible that for any two roots v_j and v_i, $\hat{R} > 1$ and thus v_j is selected by K2 as a parent of v_i when $\hat{\Gamma}$ is used, even though $R < 1$ and v_j is always rejected by K2 when Γ is used. We also note in the proofs that when $p_{\emptyset}N$ becomes larger (i.e, with more null cases in Γ), $\hat{R}/R$ becomes larger, and in turn, v_j more likely to be selected by K2 as a parent of v_i. These theorems thus provide insights and explanations for the existence of extraneous links between roots in the network structures learned by the original K2 algorithm when $p_{\emptyset}$ is large.

Since most of the learning errors made by K2 using $\hat{\Gamma}$ are extra links between roots, one way to circumvent this problem is to require that all roots be determined and specified prior to learning. These pre-specified roots thus can become parents of some non-root nodes, but not of each other. Another way to circumvent this problem is to include an appropriate number of null cases in the database. Suppose the value of $p_{\emptyset}$ is known prior to learning. Then for a non-null database $\hat{\Gamma}$ of $\hat{N}$ cases, $\beta\hat{N}$ null cases should be added to form the original Γ, where $\beta = p_{\emptyset}/(1 - p_{\emptyset})$ is the ratio of null and non-null patterns in Γ. Physically putting these null cases into the database will increase the size of the database, and thus increasing the learning time. After carefully examining the K2 algorithm, we have found that the same effect can be achieved by a minor revision of function $g(.)$ without requiring the physical insertion of null cases into $\hat{\Gamma}$.

Recall that the K2 algorithm uses function $g(i, \pi_i)$ during learning to determine the merit of adding a new node into the parent set π_i of node v_i. This function uses N_{ijk} ($k = 0$ or 1), the numbers that count the cases in Γ in which π_i takes the j^{th} of the q_i unique instantiations and v_i itself has the value zero or one (see Eq. (4)). It turns out that only one of these N_{ijk} involves the null cases. Without losing of generality, always let the first of the q_i unique instantiations of π_i in Γ be a null instantiation (all variables in π_i are instantiated to zero). Then by Eq. (4), it is clear that among all N_{ijk} in $g(i, \pi_i)$, regardless whether k being zero or one, only the first one ($j = 1$) may involve null cases. This is because for all others ($j > 1$) at least one of their variables in π_i must be instantiated to one. In other words, for $j > 1$, all cases counted by N_{ijk} contain non-null π_i, and thus must be in $\hat{\Gamma}$. So we have $N_{ijk} = \hat{N}_{ijk}$. For $j = 1, k = 1$, since cases counted by N_{i11} contain $v_i = 1$, they are not null cases, either, and thus $N_{i11} = \hat{N}_{i11}$. Therefore, all null cases are counted in

N_{i10}. However, not every case in Γ counted in N_{i10} is a null case. This is because N_{i10} only indicates that these cases have $v_i = 0$ and $\pi_i = \mathbf{0}$ but says nothing about the values the other variables take in these cases. Therefore, N_{i10} can be seen as a sum of two numbers, $\hat{N}_{i10}$, the number of *non-null* cases which contain $v_i = 0$, $\pi_i = \mathbf{0}$, and some other variables instantiated to one, and $\beta\hat{N}$, the number of all *null* cases. We can thus re-write function g, called g_β, as

$$g_\beta(i, \pi_i) = \frac{(\hat{N}_{i10} + \beta\hat{N})!\hat{N}_{i11}!}{(\hat{N}_{i1} + \beta\hat{N} + 1)!} \prod_{j=2}^{q_i} \frac{\hat{N}_{ij0}!\hat{N}_{ij1}!}{(\hat{N}_{ij} + 1)!}. \tag{11}$$

An additional experiment was conducted to test separately how K2 would perform when 1) all roots of the network were specified prior to learning and 2) different estimates of the number of null cases (measured by $\beta\hat{N}$) were provided. In the latter case, $g(.)$ was replaced by $g_\beta(.)$. In this experiment, for each of the two target networks, only one $\hat{\Gamma}$ from the previous experiment that was generated with high $p_\emptyset$ was used. For the 10 by 10 bipartite network, we chose $p_\emptyset = 0.623$ which yields $\beta = 1.651$. For the Alarm network, we chose $p_\emptyset = 0.513$ with $\beta = 1.053$. The experiment results are given in Table 2 below. The results show that when all roots were specified, no learning error occurred for either network. They also show that, without specifying the roots, g_β alone was quite effective in reducing learning errors. This holds even when the estimated β is allowed to vary $\pm 20\%$ from its true value (see rows with $\beta+$ and $\beta-$ in the table). However, we note that each of the two suggested extensions to K2 requires certain prior knowledge of the underlying domain, which may not be available in some applications.

Table 2 is about here

4 Learning with Noisy Case Data

Different kinds of noise may exist in realistic data. In this paper, we only investigate noisy data resulted from a very simple noise model. In this model, the noisy database Γ^* is generated from a given noise-free database Γ by randomly inverting each V_i in each $V \in \Gamma$ with a given fixed probability p. The noisy database Γ^* can be depicted by the following. Let ξ_i be a binary random variable with $P(\xi_i = 1) = p$. Then $v_i^* = (v_i + \xi_i) \bmod 2$. Assume that ξ_i is independent of v_i and of $\xi_j, i \neq j$. Then v_i^* in Γ^* has the probability

$$\begin{aligned} P(v_i^* = 1) &= P(v_i = 1, \xi_i = 0) + P(v_i = 0, \xi_i = 1) \\ &= p_i(1-p) + (1-p_i)p \\ &= p_i + p - 2pp_i. \end{aligned} \tag{12}$$

The added noise, if sufficiently high, may change the interdependence between variables and in turn the learned network structures. For example, suppose v_j and v_i are not directly causally related and thus conditionally independent (given π_i) to each other in Γ. This conditional independence between v_j and v_i can easily be destroyed by the noise in the case database, resulting in extra causal links in the learned network. Similarly, the noise may significantly weaken the dependency between two variables which are directly causally associated but the causal strength is weak, resulting in missing a genuine causal link. Table 3 below gives learning results of both K2 and EHL methods from two noisy databases, one for each of the two target networks. Each database contained 10,000 noisy cases with the noise level $p = 0.02$. The cut-off value was 0.01 for EHL. It can be seen that, with such a small noise, either method missed few, if any, genuine causal links. This was partly due to the relatively large c_{ji} in the two target networks. However, both methods, especially EHL, generated large numbers of extra causal links. Significantly more extra links may be generated by EHL if the cut-off is set to a lower value, say 0.005.

Table 3 is about here

We first investigate the reasons for EHL's poor performance with noisy data. Recall that with sufficiently large Γ and sufficiently many iterations, y_{ji} for all $j < i$ can be trained by the EHL method to approach their target causal strengths c_{ji}. In particular, if $c_{ji} = 0$ (i.e., there is no causal link from v_j to v_i), then $y_{ji} \to 0$. This is why an arbitrarily small cut-off value can be used to determine if a causal link exists between any pair of nodes in the learned networks. However, when a noisy database Γ^* is used, y^*_{ji} does not approach zero, even when the size of the database goes to infinity (see Theorem 4 below). As a result, a small cut-off value may lead to a large number of extraneous links.

To formally establish this, consider a pair of variables v_j and v_i where $j < i$ and $c_{ji} = 0$ in the target network, and a set of cases which share the following instantiation pattern for variables v_1 through v_i: $(v_j = 1, v_i = 1, \forall_{k<i,k\neq j} v_k = 0)$. Note that in such cases all actual parents of v_i are instantiated to zero because v_j is not a parent of v_i ($c_{ji} = 0$) and all other predecessors of v_i are zero. When there is no noise, then the probability of this set of cases in the underlying target network is zero because the probability

$$P(v_j = 1, v_i = 1, \forall_{k<i,k\neq j} v_k = 0) = 0.$$

This is because when all parents of v_i are zero in a case, v_i itself can only be zero[12]. As a consequence, cases containing such an instantiation pattern may never occur in Γ. However, it can be shown that when noise is added, the corresponding probability

$$P(v^*_j = 1, v^*_i = 1, \forall_{k<i,k\neq j} v^*_k = 0) > 0. \tag{13}$$

This result comes from the fact that many cases in Γ can be contaminated into this pattern. For example, cases with $(v_k = 0, \forall^i_{k=1})$, include, among other things, all null cases, and thus have non-zero probability. Such cases can be turned to $(v^*_j = 1, v^*_i = 1, \forall_{k<i,k\neq j}\ v^*_k = 0)$ by the following random noise of non-zero probability: $(\xi_j = 1, \xi_i = 1, \forall_{k<i,k\neq j} \xi_k = 0)$. With Eq. (13) we can establish the following.

Theorem 4. For all j, i, if $j < i$ and $c_{ji} = 0$, then y^*_{ji} does not converge to zero when Γ^* is used.

Proof: This is proved by showing that $y^*_{ji}(t) < \alpha$ does not hold for all t when $t \to \infty$, where $0 < \alpha < 1$ is the learning rate. Since, by Eq. (13), $P(v^*_j = 1, v^*_i = 1, \forall_{k<i,k\neq j}\ v^*_k = 0) > 0$, it is certain that noisy cases V^* containing $(V^*_j = 1, V^*_i = 1, \forall_{k<i,k\neq j}\ V^*_k = 0)$ exist in Γ^* if $|\Gamma^*|$ is sufficiently large,

and these cases will be applied to the learning process at some time t. When this occurs, since v_j^* is the only variable preceding v_i^* with value 1 in $V(t)$, then, by Eq. (7)

$$q_{ji}^*(t) = \frac{y_{ji}^*(t-1)}{1 - \prod_{k<i}(1 - y_{ki}^*(t-1)V_k^*(t))} = \frac{y_{ji}^*(t-1)}{y_{ji}^*(t-1)} = 1.$$

Then, regardless of the current values of $y_{ji}^*(t-1)$, we have by Eq. (6),

$$\begin{aligned} w_{ji}^*(t) &= w_{ji}^*(t-1) + \alpha(1 \cdot V_j^*(t) \cdot V_i^*(t) - w_{ji}^*(t-1)) = w_{ji}^*(t-1) \cdot (1-\alpha) + \alpha > \alpha, \text{ and} \\ y_{ji}^*(t) &= w_{ji}^*(t)/x_i(t) > w_{ji}^*(t) > \alpha, \text{ since } x_i^*(t) < 1 \ \forall t. \end{aligned}$$

□

From Theorem 4, causal strengths which converge to zero with large noise-free Γ will not converge with large noisy Γ^*. As a result, many extraneous spurious causal links can be included in the learned network when the cut-off value used is arbitrarily small. This explains the existence of a large number of extra links in the learned networks in the experiment (Table 3). Therefore, a simple way to reduce the number of extra links induced by noise is to raise the cut-off value. It is conceivable that higher cut-off values may mistakenly exclude some genuine causal links of small learned causal strengths, leading to an increase in the number of missing links. To see the effect of higher cut-off values on the overall learning quality, we repeated the previous experiment (the noise rate $p = 0.02$) with different cut-offs for the two target networks. The results are reported in Table 4 below. As expected, when the cut-off value increases, the number of extra links decreases and the number of missing links increases.

Table 4 is about here

This simple approach to cope with noise in case data has a number of limitations. First, it lacks the guidance for selecting appropriate cut-off values for applications in different domains and with unknown noise levels. Also, it might be inappropriate to apply a single cut-off value to the entire network. This is because for different nodes the causal strengths from their parents may have quite different magnitudes. A single or global large cut-off value may work well with nodes whose incoming causal links all have high strengths but not with other nodes with weak incoming causal strengths. To cope with these problems, we propose another way to deal with the noisy data. Recall that with Eq. (7), for a given v_i, all learned causal strengths $y_{ki}(t)$ for all $k < i$ compete to each other when a new case input is presented, with individual learned strengths y_{ki} as the competition factor. In other words, the local competition among potential parents of a node determines the final fate of these links. Also note that since the extra links induced by noise often have small causal strengths, they may be suppressed by genuine links if this local competition can be increased. One way to increase the competition, borrowed from the study of competitive neural networks[15], is to use an exponent of y_{ji} rather than y_{ji} itself in Eq. (7) as follows.

$$\tilde{q}_{ji}(t) = \frac{y_{ji}^{c}(t-1)}{1 - \prod_{k<i}(1 - y_{ki}^{c}(t-1)V_k(t))}, \tag{14}$$

where $c \geq 1$ is the competition rate. When $c = 1$, $\tilde{q}_{ji}$ of Eq. (14) is reduced to q_{ji} of Eq. (7). The previous experiment with noisy case data was repeated with low cut-off value (0.01) and increased local competition ($c = 1.2$). As can be seen in Table 4, the numbers of missing and extra links were reduced to 1 and 1 for the 10x10 network, and 3 and 2 for the Alarm belief network, respectively. This is a significant improvement comparing with the results using larger, global cut-off values in the same table.

Next we investigate the behavior of K2 algorithm with noisy data. We noted from Table 3 that, comparing to EHL method, K2 adds fewer extra causal links to the resulting network. This suggests that even though K2 was not developed for noisy data, it has some inherent noise resistance capability. To see this more clearly, consider a theorem given by Bouckaert[1] (re-produced as Theorem 5 below) in his analysis of belief network learning using the minimum description length (MDL) principle. This theorem can be proven by straightforward applications of Sterling's formula. The actual proof can be found in[2].

Theorem 5: Let $Q(B_S, \Gamma)$ be the quality of a network structure B_S learned from database Γ defined by

$$Q(B_S, \Gamma) = \log P(B_S) - N \cdot H(B_S, \Gamma) - \frac{1}{2} h \log N \tag{15}$$

where $h = \sum_{i=1}^{n} q_i(r_i - 1)$ and $H(B_S, \Gamma) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} -\frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}}$, and all other terms are the same as defined in Eq. (3). Then

$$\log P(B_S, \Gamma) = Q(B_S, \Gamma) + O(1) \tag{16}$$

with respect to N, the total number of cases in Γ. □

This theorem connects the Bayesian measure of the quality of a learned structure, namely $P(B_S, \Gamma)$, with an MDL measure $Q(B_S, \Gamma)$. If the prior probabilities of all structures are assumed to be equal, then $Q(B_S, \Gamma)$ is determined by two terms. The term $H(B_S, \Gamma)$ represents the conditional entropy of B_S. The more information is given in B_S the lower the entropy is. Therefore, adding causal links to a node will decrease the entropy term $H(.)$, and in turn increase $Q(.)$. On the other hand, the term $(h \log N)/2$, called parsimony term, is induced by the principle of Occam's razor: structures with fewer links are preferred to those with more links. As such, when more links are added during learning, this term becomes larger, and thus $Q(.)$ becomes smaller. The quantity Q thus seeks a balance between the two opposite demands, namely to introduce more links to maximize the fitness of the structure to the data and fewer links to minimize the complexity of the structure.

The point that interests us is, since $P(B_S, \Gamma)$ and $Q(B_S, \Gamma)$ differ only by $O(1)$ when N is large, this theorem suggests that the Bayesian measure $P(B_S, \Gamma)$ used in K2 does contain a component similar to the parsimony term $(h \log N)/2$ that prevents the learned structure from becoming arbitrarily complex. This may explain the noise-resistance capability exhibited by K2 in the experiment. Moreover, this capability can be amplified if we can increase the weight of this term during learning. This can be done by replacing $g(i, \pi_i)$ by $\log \breve{g}(i, \pi_i) = \log g(i, \pi_i) - \gamma h \log N$ where γ is the weight of the added parsimony term. One problem with this approach is that the direct computation of h involves computing q_k for all nodes $v_k$in the network. This computation is expensive because it is global ($\forall k$) and dynamic (each step of learning changes one π_i and in turn its q_i). However, K2 is only interested in the direction and magnitude of the change of $\breve{g}(.)$ when v_j is added into π_i, and adding v_j into π_i does not change the value of any q_k except q_i. This leads to the following tractable local computation of the change of $\breve{g}(.)$.

$$\begin{aligned} \Delta &= \breve{g}(i, \pi_i \cup \{v_j\}) - \breve{g}(i, \pi_i) \\ &= \log g(i, \pi_i \cup \{v_j\}) - \log g(i, \pi_i) - \gamma(q_i^{\{\pi_i\}} - q_i^{\{\pi_i \cup \{v_j\}\}}) \log N \end{aligned} \tag{17}$$

where $q_i^{\{\pi_i\}}$ and $q_i^{\{\pi_i \cup \{v_j\}\}}$ are the numbers of unique patterns of π_i and $\pi_i \cup \{v_j\}$, respectively. Then, if $v_j \in \{v_1 \cdots v_n\} \backslash \pi_i$ maximizes Δ and the maximized $\Delta > 0$, v_j is added into the current π_i.

We further experimented the modified K2 algorithm described above for the two networks with noise level $p = 0.02$ and different γ. The results are reported in Table 5 below. Compared with the results in Table 3, the modified K2 is quite effective in dealing with noisy data, especially with the Alarm network.

Table 5 is about here

---

5 Additional Experiments

In the two previous sections we have suggested several extensions to both EHL and K2 learning methods to better handle realistic case data. For EHL, we have suggested to use either an increased global cut-off value or the local competition (with $c > 1$) to handle noisy case data. For the K2, we have suggested a way to include an estimated number of null cases ($\beta\hat{N}$) in learning (replacing function $g(.)$ of Eq. (4) by $\tilde{g}(.)$ of Eq. (14)) to handle non-null case data, and a way to introduce into learning an additional MDL factor (a term in function $\breve{g}(.)$ of Eq. (17) with γ as its weight). Since these suggestions are made in isolation, their effectiveness in handling case data with both features, i.e., those comprised of *noisy non-null* cases, is not clear. To test the performance of these suggested modifications with different parameters (the cut-off values, the values for c, β, and γ), we conducted a sequence of additional computer experiments over the two belief networks.

The case data used in the experiments were prepared as follows. First, two databases of 10,000 non-null cases each were randomly generated from the two target networks with $p_\emptyset = 0.623$ and 0.513, respectively (all null cases generated during the process were discarded). Then random noise with varying noise levels of 0.02, 0.04, 0.06, 0.08, and 0.10 was added, according to the noise model described in Section 4 (Eq. (12)), to these two databases to create *noisy non-null* databases. For each noise level, ten different noisy databases were created. There were thus one noise-free non-null database and 50 noisy ones for each of the two target networks. The experiment results were averaged over the ten noisy data sets for each noise level.

For the EHL method, experiments were conducted without and with local competition ($c = 1.0$, and 1.2) and with different cut-off values ($p = 0.01, 0.05$, and 0.10). All experiments were done with the learning rate $\alpha = 0.001$ and with 100,000 iterations (i.e., each of the 10,000 cases in a database is presented 10 times to the learning program). Since EHL is not affected by the removal of null cases, we expect the two modifications for handling noisy data (increased cut-off values and local competition) to be effective when applied to noisy non-null databases. The results of these experiments, reported in Table 6 below, confirmed our expectation.

Table 6 is about here

It can be seen from the numbers before the "/" signs in Table 6 that, when no additional competition was introduced ($c = 1.0$), the number of extra links decreased much faster than the increase of the number of missing links as the cut-off value moved higher. This result empirically confirms that using a single higher cut-off value alone can increase the noise resistance of EHL. A more interesting observation is that increasing competition had the similar effect (see numbers after the "/" signs in Table 6), and its overall performance was significantly better than using a single higher cut-offs alone, especially when the noise level was low ($p = 0.02 \sim 0.06$). The best combination of the parameters for both target networks was cut-off = 0.05 and the competition rate $c = 1.2$ for these experiments.

In the experiments, the K2 method was extended to include both an estimated ratio of null cases (β in Eq. (11)) and a parsimony term (weighted by γ in Eq. (17)). Since K2 learning is not very sensitive to the actual value of β (see Table 2), we used fixed values of $\beta = 1.4$ for the 10x10 network and $\beta = 0.9$ for the Alarm network in the experiment. Both of these two estimates were about 0.8 of their respective actual values of 1.651 and 1.063. We allowed γ to vary from 0 to 16. The results are reported in Table 7 below. The results show that extra links can be suppressed by the parsimony term at the cost of a modest increase of missing links. This effect became stronger when γ became larger. The best γ value we found for these two networks was 16.

Table 7 is about here

Although the scope is limited, the experiment results suggest that both EHL and K2 methods become much more effective in handling noisy non-null data with the proposed modifications. For EHL, the best combination of parameters ($c = 1.2$ and cut-off=0.05) resultsed on average in missing 5 and adding 0.42 links for the 10 x 10 network, and 6.37 and 2.1, respectively, for the Alarm network. The corresponding numbers for the extended K2 with the best $\gamma = 16$ were 12.03 and 3.22, and 7.13 and 5.35, respectively. The results also suggest that the EHL method performed noticeably better than the K2 method with the proposed modifications

6 Conclusions

In this paper we have investigated the learning behaviors of K2 and EHL methods for constructing noisy-or belief networks from realistic data of noisy non-null cases. Both analytical and empirical evidence suggests that removing null cases from the database does not affect EHL learning, but may cause K2 to generate significant number of extra links between roots. The evidence also suggests that both methods, especially EHL, are sensitive to noise, and most learning errors with noisy data are also extra causal links. Several modifications to the two methods were suggested to cope with these problems, and they have shown to be quite effective. These suggestions thus can serve as a basis for real-world applications where case data may be noisy and some or all null cases may be excluded.

Further research is needed in developing methods that can construct belief networks from realistic but not ideal case data. The work reported in this paper is a first step along that direction, and is limited in several aspects. For example, the noise model used here is simple, and unrealistic for many applications. Also, this work is restricted to networks of binary variables. Further work is needed to extend this work to handle noisy data from other noise models, and handle multi-value variables. Study of other types of realistic data in belief network construction is also needed. Finally, the experiments reported in this paper all used artificially generated case data. It would be very beneficial if various belief network learning methods, including those proposed in this paper, can be tested using real case data collected from real-world applications of different domains.

Appendix

1. Proof of Theorem 2.

By Eq. (4), with $r_i = 2\ \forall i$ we have

$$g(i, \emptyset) = \frac{N_{v_i=0}!N_{v_i=1}!}{(N_{v_i=0} + N_{v_j=1} + 1)!},$$

where $N_{v_i=0}$ and $N_{v_i=1}$ are the numbers of cases in Γ with $v_i = 0$ and $v_i = 1$, respectively. Let p_i denote $P(v_i = 1)$ and p_j denote $P(v_j = 1)$ in Γ. Then $N_{v_i=0} = (1 - p_i)N$ and $N_{v_j=1} = p_i N$, and

$$g(i, \emptyset) = \frac{((1-p_i)N)!(p_iN)!}{(N+1)!} = \frac{1}{N+1} \cdot \frac{((1-p_i)N)!(p_iN)!}{N!}.$$

Applying Sterling formula to each of the three factorials, we have

$$g(i, \emptyset) \quad = \quad \frac{1}{N+1} \cdot \frac{\sqrt{2\pi(1-p_i)N}e^{-(1-p_i)N}((1-p_i)N)^{(1-p_i)N}\sqrt{2\pi p_i N}e^{-p_iN}(p_iN)^{p_iN}}{\sqrt{2\pi N}e^{-N}N^N}$$

$$
\begin{aligned}
&= \frac{\sqrt{2\pi p_i(1-p_i)N}}{N+1} \cdot \frac{(p_iN)^{p_iN}((1-p_i)N)^{(1-p_i)N}}{N^N} \\
&= \frac{\sqrt{2\pi p_i(1-p_i)N}}{N+1} \cdot \left[\frac{(p_iN)^{p_i}((1-p_i)N)^{(1-p_i)}}{N}\right]^N \\
&= \frac{\sqrt{2\pi p_i(1-p_i)N}}{N+1} \cdot \left[p_i^{p_i}(1-p_i)^{1-p_i}\right]^N . \qquad (18)
\end{aligned}
$$

Now consider the situation when $\pi_i = \{j\}$ where v_j and v_i are independent of each other. There are only two unique instantiations of π_i in this case, namely, $v_j = 0$ and $v_j = 1$. Then, by Eq. (4) again,

$$
g(i, \{j\}) = \frac{N_{v_j=0,v_i=0}!N_{v_j=0,v_i=1}!}{(N_{v_j=0}+1)!} \cdot \frac{N_{v_j=1,v_i=0}!N_{v_j=1,v_i=1}!}{(N_{v_j=1}+1)!}.
$$

Since v_i and v_j are independent of each other, we have $N_{v_j=0,v_i=0} = (1-p_j)(1-p_i)N$, $N_{v_j=0,v_i=1} = (1-p_j)p_iN$, $N_{v_j=0} = (1-p_j)N$; $N_{v_j=1,v_i=0} = p_j(1-p_i)N$, $N_{v_j=1,v_i=1} = p_jp_iN$, $N_{v_j=1} = p_jN$. Let $N' = (1-p_j)N, N'' = p_jN$. Then, analogous to the derivation of Eq. (18), we have,

$$
\begin{aligned}
\frac{N_{v_j=0,v_i=0}!N_{v_j=0,v_i=1}!}{(N_{v_j=0}+1)!} &= \frac{((1-p_i)N')!(p_iN')!}{(N'+1)!} \\
&= \frac{\sqrt{2\pi p_i(1-p_i)N'}}{N'+1} \left[p_i^{p_i}(1-p_i)^{1-p_i}\right]^{N'} . \qquad (19)
\end{aligned}
$$

$$
\begin{aligned}
\frac{N_{v_j=1,v_i=0}!N_{v_j=1,v_i=1}!}{(N_{v_j=1}+1)!} &= \frac{((1-p_i)N'')!(p_iN'')!}{(N''+1)!} \\
&= \frac{\sqrt{2\pi p_i(1-p_i)N''}}{N''+1} \cdot \left[p_i^{p_i}(1-p_i)^{1-p_i}\right]^{N''} . \qquad (20)
\end{aligned}
$$

Combining Eqs. (18), (19), and (20),we then have the ratio

$$
\frac{g(i,\{j\})}{g(i,\emptyset)} = \frac{N+1}{(N'+1)(N''+1)} \cdot \frac{\sqrt{2\pi p_i(1-p_i)N'} \cdot \sqrt{2\pi p_i(1-p_i)N''}}{\sqrt{2\pi p_i(1-p_i)N}} \cdot \left[(p_i^{p_i}(1-p_i)^{1-p_i}\right]^{N'+N''-N} .
$$

Note that $N' + N'' = (1-p_j)N + p_jN = N$ and that $(N' \cdot N'')/N = (p_j(1-p_j)N)$, then,

$$
\begin{aligned}
\frac{g(i,\{j\})}{g(i,\emptyset)} &= \sqrt{2\pi p_i(1-p_i)p_j(1-p_j)} \cdot \sqrt{N} \cdot \frac{N+1}{((1-p_j)N+1)(p_jN+1)} \\
&= \sqrt{2\pi p_i(1-p_i)p_j(1-p_j)} \cdot \sqrt{N} \cdot \frac{N+1}{p_j(1-p_j)N^2+N+1} \\
&< \sqrt{2\pi p_i(1-p_i)p_j(1-p_j)} \cdot \sqrt{N} \cdot \frac{N+1}{p_j(1-p_j)(N+1)N} \\
&= \sqrt{\frac{2\pi p_i(1-p_i)}{p_j(1-p_j)}} \cdot \frac{1}{\sqrt{N}} \\
&< 1, \qquad (21)
\end{aligned}
$$

where the first inequality comes from the fact that $p_j(1-p_j)(N+1)N < p_j(1-p_j)N^2+N+1$, and the last inequality holds when N is allowed to be arbitrarily large. $\square$

2. Proof of Theorem 3.

By the definition of $g(.)$ and $\hat{g}(.)$, we have $g(i,\emptyset) = (N_{v_i=0}!N_{v_i=1}!)/(N+1)!$ and $\hat{g}(i,\emptyset) = (\hat{N}_{v_i=0}!\hat{N}_{v_i=1}!)/(\hat{N}+1)!$. Since removal of null patterns does not affect the case counts which involve non-null instantiations, then $\hat{N}_{v_i=1} = N_{v_i=1}$, and

$$\frac{g(i,\emptyset)}{\hat{g}(i,\emptyset)} = \frac{N_{v_i=0}!}{\hat{N}_{v_i=0}!}\frac{(\hat{N}+1)!}{(N+1)!}. \tag{22}$$

Similarly, we have

$$\frac{\hat{g}(i,\{j\})}{g(i,\{j\})} = \frac{\hat{N}_{v_j=0,v_i=0}!}{N_{v_j=0,v_i=0}!}\frac{((N_{v_j=0}+1)!}{(\hat{N}_{v_j=0}+1)!}. \tag{23}$$

As shown in the proof of Theorem 2, we have $N_{v_i=0} = (1-p_i)N$, $N_{v_j=0} = (1-p_j)N$, and $N_{v_j=0,v_i=0} = (1-p_j)(1-p_i)N$. Let $b = 1-p_\emptyset$. Then $\hat{N} = bN$. To process the other factorials, we need the following conditional probabilities:

$$P(v_i = 1|\hat{\Gamma}) = P(v_i = 1)/(1-p_\emptyset) = p_i/b;$$

$$P(v_j = 1|\hat{\Gamma}) = p_j/b;$$

$$P(v_j = 1, v_i = 1|\hat{\Gamma}) = p_jp_i/b.$$

Noting that v_j and v_i are independent of each other in Γ but not in $\hat{\Gamma}$, then

$$\hat{N}_{v_i=0} = P(v_i = 0|\hat{\Gamma}) \cdot \hat{N} = (1-p_i/b) \cdot bN = (b-p_i)N;$$

$$\hat{N}_{v_j=0} = (1-p_j/b) \cdot bN = (b-p_j)N;$$

$$\hat{N}_{v_j=0,v_i=0} = (1-p_j/b-p_i/b+p_jp_i/b) \cdot bN = (b-p_j-p_i+p_jp_i)N \text{ and}$$

$$N_{v_j=0,v_i=0} = (1-p_j-p_i+p_jp_i) \cdot N.$$

Substituting the above into Eqs. (22) and (23), we have

$$\begin{aligned}\frac{\hat{R}}{R} &= \frac{\hat{g}(i,\{j\}) \cdot g(i,\emptyset)}{g(i,\{j\}) \cdot \hat{g}(i,\emptyset)} \\ &= \frac{((1-p_i)N)!}{((b-p_i)N)!} \cdot \frac{(bN+1)!}{(N+1)!} \cdot \frac{((b-p_j-p_i+p_jp_i)N)!}{((1-p_j-p_i+p_jp_i)N)!} \cdot \frac{((1-p_j)N+1)!}{((b-p_j)N+1)!}\end{aligned}$$

$$
\begin{aligned}
= & \frac{(1-p_i)N\cdots((b-p_i)N+1)}{(1-p_i-p_j+p_ip_j)N\cdots((b-p_i-p_j+p_ip_j)N+1)}\cdot \\
& \frac{((1-p_j)N+1)\cdots((b-p_j)N+2)}{(N+1)\cdots(bN+2)}.
\end{aligned} \tag{24}
$$

Note that each of the four terms in Eq. (24) is a product of $(1-b)N$ consecutive integers. For brevity, we write Eq. (24) as

$$
\frac{\hat{R}}{R} = \prod_{k=1}^{(1-b)N} \frac{A_{1,k}A_{2,k}}{B_{1,k}B_{2,k}}. \tag{25}
$$

The first factor in the product

$$
\frac{A_{1,1}A_{2,1}}{B_{1,1}B_{2,1}} = \frac{(1-p_i)N\cdot((1-p_j)N+1)}{(1-p_i)(1-p_j)N\cdot(N+1)} = \frac{(1-p_j)N+1}{(1-p_j)N+(1-p_j)} > 1. \tag{26}
$$

The $(k+1)^{th}$ factor in the product

$$
\begin{aligned}
\frac{A_{1,k+1}A_{2,k+1}}{B_{1,k+1}B_{2,k+1}} &= \frac{(A_{1,1}-k)(A_{2,1}-k)}{(B_{1,1}-k)(B_{2,1}-k)} \\
&= \frac{A_{1,1}A_{2,1}-k(A_{1,1}+A_{2,1})+k^2}{B_{1,1}B_{2,1}-k(B_{1,1}+B_{2,1})+k^2}.
\end{aligned} \tag{27}
$$

Note that

$$
A_{1,1}+A_{2,1} = (1-p_i)N+(1-p_j)N+1 = 2N-(p_i+p_j)N+1;
$$

$$
B_{1,1}+B_{2,1} = (1-p_i-p_j+p_ip_j)N+N+1 = 2N-(p_i+p_j)N+1+p_ip_jN.
$$

So $A_{1,1}+A_{2,1} < B_{1,1}+B_{2,1}$. Combining this and $A_{1,1}A_{2,1} > B_{1,1}B_{2,1}$ (Eq. (26)) with Eq. (27), we have $A_{1,k+1}A_{2,k+1} > B_{1,k+1}B_{2,k+1}$ for $k > 1$. Then by Eq. (25), $\hat{R}/R > 1$. Moreover, since each of the $(1-b)N$ terms in the product of Eq. (25) is greater than 1, it can be concluded that if all things remain the same, then the larger $(1-b)N = p_\emptyset N$ is, the higher the ratio $\hat{R}/R$ will be. $\square$

References

1. R. Bouckaert, "Probabilistic Network Construction Using the Minimum Description Length Principle", in *Proceedings ECSQARU*, 41-48, 1993.

2. R. Bouckaert, "Belief network Construction Using the Minimum Description Length Principle", Technical Report RUU-CS-93-??. Utrecht University, The Netherlands, 1992.

3. G. Cooper and E. Herskovits, "A Bayesian Method for the Induction of Probabilistic Networks from Data", *Machine Learning*, **9**:309-347, 1992.

4. D. Hebb, *The Organization of Behavior, a Neuropsychological Theory*, New York: John-Wiley, 1949.

5. D. Heckerman, D. Geiger, and D. Chickering, "Learning Bayesian Networks: The Combination of Knowledge and Statistics", in *Uncertainty in artificial Intelligence, The Proceedings of the Tenth Conference*, R. L. de Mantaras and D. Poole (eds.), 293-301, 1994.

6. G. Hinton,"Connectionist Learning Procedures", *Artificial Intelligence*, **40**, 185-234, 1989.

7. W. Lam and F. Bacchus, "Learning Bayesian Belief networks: An Approach Based on the MDL Principle", *Computational Intelligence*, **10**, 269-293, 1994.

8. R Neal, "Connectionist learning of belief networks", *Artificial Intelligence*, **56**, 71-113, 1992.

9. R. Neapolitan, *Probabilistic Reasoning in Expert Systems*, New York: John Wiley & Sons, 1990.

10. J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Calif.: Morgan Kaufmann Publishers, 1988.

11. J. Pearl and T. Verma, "A Theory of Inferred Causation", in *Principles of Knowledge Representation and Reasoning: Proc. of the Second Intl. Conf.*, 441-452, 1991.

12. Y. Peng and J. Reggia, *Abductive Inference Methods for Diagnostic Problem-Solving*, New York: Springer-Verlag, 1990.

13. Y. Peng, "Learning Probabilities for Causal networks", in *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, MD, June, **IV**, 97-102, 1992.

14. Y. Peng and Z. Zhou, "A Neural Network Learning Method for Belief Networks", *International Journal of Intelligent Systems*, **11**, 893-916, 1996.

15. J. Reggia, Y. Peng, and P. Bourret, "Recent Application of Competitive Activation Mechanisms", in *Neural Network: Advances and Applications*, E. Gelenbe (ed.), 33-62, Amsterdam: North-Holland, 1991.

16. P. Spirtes, C. Glymour, and R. Scheines, “An Algorithm for Fast Recovery of Sparse causal Graphs”, *Social Science Computer Review*, **9**, 62-72, 1991.

17. J. Suzuki, “A Construction of Bayesian Networks from Database Based on An MDL Scheme”, in *Uncertainty in Artificial Intelligence, Proceedings of the Ninth Conference*, D, Heckerman and A. Mamdani (eds.), 266-273, 1993.

Captions for Figures

Figure 1 A 10 $\times$ 10 bipartite belief network.

Figure 2 The Alarm belief network structure, consisting of 37 nodes and 46 links.

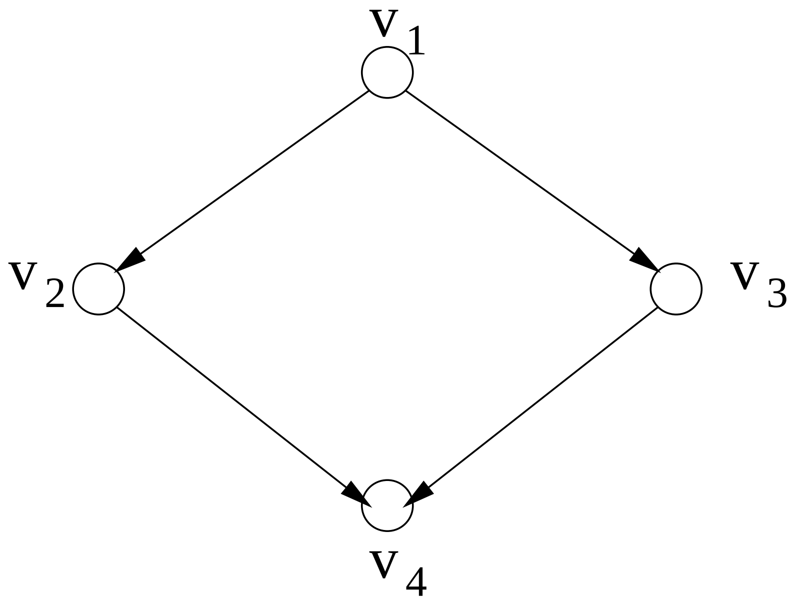

Figure 1.

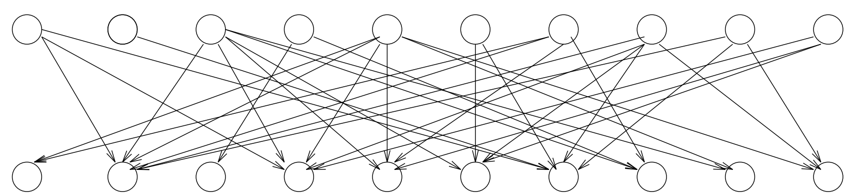

Figure 2.

Captions for Tables

Table 1 Learning from non-null cases by K2 and EHL.

Table 2 Learning from non-null cases by extended K2.

Table 3 Learning from noisy cases by K2 and EHL.

Table 4 EHL learning from noisy cases with different cut-offs and local competition

Table 5 K2 learning from noisy cases with different γ

Table 6 Learning from noisy non-null data by extended EHL. Numbers before and after "/" are for $c = 1.0$ and $c = 1.2$, respectively.

(a) 10 x 10 bipartite

(b) Alarm network

Table 7 Learning from noisy non-null data by extended K2.

(a) 10 x 10 bipartite network

(b) Alarm network

Table 1.

target network	$p_\emptyset$	K2		EHL	
		missing	extra	missing	extra
10 x 10	0.623	0	44	0	1
bipartite	0.063	0	14	0	0
network	0.0064	2	1	0	0
Alarm	0.513	0	59	0	1
network	0.046	0	9	0	2
	0.003	0	0	0	1

Table 2.

target network	prior knowledge	missing	extra
10 x 10 bipartite network	roots specified	0	0
	$\beta = 1.651$	0	1
	$\beta- = 1.321$	0	3
	$\beta+ = 1.981$	0	3
Alarm network	roots specified	0	0
	$\beta = 1.053$	0	1
	$\beta- = 0.843$	0	3
	$\beta+ = 1.264$	0	0

Table 3.

target network	K2 algorithm		EHL method	
	missing	extra	missing	extra
10 x 10 bipartite network	0	14	0	26
Alarm network	2	35	2	70

Table 4.

target network	cut-off/c	missing	extra
10 x 10 bipartite network	0.01/1.0	0	26
	0.05/1.0	2	7
	0.10/1.0	5	3
	0.01/1.2	1	1
Alarm network	0.01/1.0	2	70
	0.05/1.0	4	8
	0.10/1.0	7	0
	0.01/1.2	3	2

Table 5.

target network	γ	missing	extra
10 x 10	0	0	14
bipartite	2	2	9
network	4	8	5
Alarm	0	2	35
network	2	5	11
	4	5	9

Table 6.

(a)

p	$cut\_off = 0.01$		$cut\_off = 0.05$		$cut\_off = 0.10$	
	missing	extra	missing	extra	missing	extra
0.00	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0
0.02	0.0/0.2	10.7/0.5	0.0/0.2	4.9/0.2	2.0/0.5	1.7/0.1
0.04	0.0/0.3	15.4/0.3	0.2/0.3	7.4/0.0	5.5/3.1	3.7/0.0
0.06	0.0/3.0	18.8/1.5	1.9/3.7	7.9/0.4	7.2/8.3	5.2/0.1
0.08	0.5/9.7	19.7/1.4	3.3/10.4	8.7/0.5	11.0/12.5	5.0/0.0
0.10	0.9/15.4	20.0/2.9	6.0/15.4	9.2/1.4	13.6/16.5	5.1/0.5
$mean$	0.23/4.77	14.1/1.1	1.9/5.0	6.4/0.42	6.55/6.82	3.45/0.12

(b)

p	$cut-off = 0.01$		$cut\_off = 0.05$		$cut\_off = 0.10$	
	missing	extra	missing	extra	missing	extra
0.00	0.0/0.0	0.0/1.0	1.0/2.0	0.0/0.0	1.0/3.0	0.0/0.0
0.02	0.1/1.2	35.5/0.8	2.2/3.1	4.6/0.4	5.6/5.6	0.0/0.1
0.04	0.5/4.2	61.8/3.3	3.8/5.4	10.4/0.8	8.8/7.9	0.7/0.4
0.06	0.8/7.1	80.9/10.3	4.4/7.5	16.6/2.1	11.2/10.6	2.7/1.6
0.08	2.5/7.8	92.0/10.5	5.9/8.6	20.2/2.6	13.0/12.0	4.4/1.3
0.10	2.9/11.0	107.5/26.0	7.0/11.6	24.1/6.7	15.1/14.0	4.0/3.0
$mean$	1.13/5.22	62.95/8.65	4.05/6.37	12.65/2.1	9.12/8.85	1.97/1.07

Table 7.

(a)

pp	$\gamma = 0$		$\gamma = 4$		$\gamma = 8$		$\gamma = 16$	
	missing	extra	missing	extra	missing	extra	missing	extra
0.00	0.0	0.0	0.0	1.0	2.0	1.0	8.0	1.0
0.02	0.0	47.4	1.4	1.9	7.2	1.3	10.0	1.3
0.04	0.0	77.5	5.0	9.5	7.9	2.3	12.4	1.8
0.06	0.4	88.0	5.8	20.3	8.6	10.3	13.7	2.1
0.08	0.6	92.9	7.0	28.5	9.0	15.0	14.0	2.5
0.10	1.2	95.6	8.0	33.0	10.1	22.4	14.1	10.6
mean	0.37	66.9	4.53	15.1	7.47	8.72	12.03	3.22

(b) Alarm network

pp	$\gamma = 0$		$\gamma = 4$		$\gamma = 8$		$\gamma = 16$	
	missing	extra	missing	extra	missing	extra	missing	extra
0.00	0.0	0.0	0.0	0.0	1.0	0.0	5.0	0.0
0.02	0.3	70.2	2.4	6.7	5.1	0.4	6.1	0.1
0.04	1.3	125.5	4.4	12.5	5.3	6.2	5.5	1.8
0.06	2.4	160.8	6.2	27.9	7.2	10.3	7.5	5.9
0.08	3.2	182.0	6.7	39.7	7.7	19.2	8.9	7.5
0.10	4.5	198.9	7.9	50.4	8.5	24.9	9.8	16.8
mean	1.95	122.9	4.6	22.87	5.8	10.17	7.13	5.35