A Neural Network Learning Method for Belief Networks

Yun Peng* and Zonglin Zhou†

Department of Computer Science

University of Maryland Baltimore County

Baltimore, MD 21228

[1] **Abstract**

This paper presents a learning method for a special class of belief networks known as noisy-or networks. By extending the Hebbian rule of neural networks, two learning rules are developed to learn the probabilities of nodes and the inter-node causal strengths, respectively. The latter rule also learns structures of networks because a non-zero causal strength indicates the existence of a causal link. One distinct feature of this method is its ability to work in a sequential or incremental manner in which a network adjusts its parameters upon the arrival of every case description. As a result, this method is capable of not only constructing a causal knowledge base from a fixed set of case data but also dynamically adapting an existing knowledge base to a changing environment. To prove the convergence of learning, a Liapunov function is identified for the dynamic system defined by the learning rules. Computer experiment results show that this method is significantly faster than some existing learning methods for belief networks.

1 Introduction

Bayesian Belief networks distinguish themselves from other AI and statistical models for probabilistic and causal inferences by their naturalness and expressiveness in representing domain causal knowledge and their rigorous treatment of uncertainty of causal associations using the formal Bayesian probability theory[20,22,19]. However, their applications to real-world problems are restricted by, among other things, the lack of effective methods to acquire the needed causal knowledge[5].

One approach to this problem is machine learning, especially learning causal knowledge from large amount of case data. This approach is especially attractive as a knowledge acquisition method

[1]* This work was supported in part by NSF Grant IRI-9309136 to Yun Peng

† On leave from The Institute of Computing Technology, Chinese Academy of Sciences.

in fields such as medicine where huge amount of case data are available[2]. Active research work has been conducted along this line in recent years. Examples of recent development include methods which infer a causal network structure that best fits the given database of cases in terms of conditional dependency between variables (e.g., IC-algorithm of Pearl and Verma[21], PC-algorithm of Spirtes, Glymour and Scheines[23]); Bayesian methods which construct a causal network with the most probable structure, given a database of cases (e.g., the K2 algorithm of Cooper and Herskovits[6], and the work of Heckerman and Chickering that combines prior knowledge and case data in Bayesian learning[9]); and methods using minimum description length, a measure of how well a causal structure fits the given data based on cross entropy[15,24]. The learning process under these methods can be categorized as *non-incremental* in that the learning decision at each step involves inspection of many, sometime all cases in an existing database. Consequently, the case database must be fixed during the whole learning process, making it difficult for these methods to handle applications where a knowledge base needs to be frequently updated by newly arrived case data. Moreover, since all of these methods involve searching the space of all possible network structures for the one which best fits the given data, their time complexities are, in general, exponential to the network size when exact solutions are sought. The complexity remains high even when heuristic search is used for approximate solutions. For example, the K2 algorithm has a complexity of $O(mn^4)$, where m is the number of cases in the database and n the number of nodes in the network[6].

A large collection of learning methods has been developed in neural network studies. Different from more traditional machine learning, learning in neural networks involves adjusting, based on case data, the connection weights so that the network displays some desired behavior. Since both neural networks and belief networks are networks of nodes and weighted links, adopting some neural network learning methods to belief networks is particularly appealing. Many researchers have used *supervised* learning methods such as backpropagation to learn problem-solving knowledge from case data for causal inferences such as diagnosis[1,3,12,17]. However, the learning results from these methods are not belief networks with application-independent causal knowledge but neural network parameters aimed at best performing the specific task the network is trained to perform. A noticeable exception is Neal's method which applies the Boltzmann machine model, an *unsupervised* learning of neural networks to belief networks[18]. In this approach, gradient ascent method is used to construct a belief network which will maximize the similarity between the probability distributions of the variables in the data and in the network. One problem with this method is that the learning takes long time to complete. Also, the learning network may get stuck at local but not global maximum of the likelihood[14,18]. Simulated annealing may be used to deal with this problem, but the time required for learning would be further increased significantly[10,18].

The Hebbian rule[8,10,14], a general unsupervised neural network learning method, has been known to be good in learning associations between variables of interest. In essence, this rule increases the synaptic connection weight between two neurons whenever both of them are activated. Weights learned by the Hebbian rule thus reflect the strengths of associations between nodes or variables in the case data without relating to any specific application task. The principle of Hebbian rule can be applied to learning probabilities. For example, Levy and Burger derived a Hebbian-like method to learn conditional probability $P(x|y)$ from case data when they were developing a model for synaptic modification for contralateral and ipsilateral synapses[16]. However, since Hebbian learning is based on co-occurrence of variables, it may not be directly applicable to learning causal relations.

In this paper we present a new method by modifying the Hebbian rule toward learning both structures and probability distributions of a special class of belief networks known as *noisy-or networks*. Two learning rules are developed for learning prior probabilities of nodes and inter-node causal strengths, respectively. The latter rule also learns the structures of networks because non-zero causal strengths indicate the existence of causal links. Learning is done by updating the variables of the respective probabilities *whenever* a case description is applied, and thus is *sequential* or *incremental*. By utilizing special features of noisy-or networks, this method is more efficient than other methods that are developed for general belief networks. The rest of this paper is organized as follows. Section 2 briefly describes the formulation of the noisy-or belief networks. Section 3 presents and analyzes the two learning rules. Computer experiments with two noisy-or belief networks are described in Section 4. Section 5 compares this new method with Neal's Boltzmann machine method and the K2 algorithm of Cooper and Herskovits. Section 6 concludes with suggestions for future research.

2 Noisy-Or Belief Networks

In the most general form, a belief network consists of a directed acyclic graph (DAG) of nodes and arcs, referred to as network structure, and a conditional probability distribution (CPD) associated with the graph[20,19]. A node in DAG corresponds to a variable v_i. In this paper, we restrict all variables to *binary*, and use $v_i = 1$ to denote that v_i is true, present, or occurring, in a given case; and $v_i = 0$ for the opposite. Also, taking the causal interpretation of the arcs, we consider an arc $< v_i, v_j >$ as a causal link representing the existence of a direct cause-effect association from v_i to v_j. The CPD consists of $P(v_i|\pi_i)$, the conditional probability 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, $P(v_i|\pi_i)$ becomes $P(v_i)$, the prior probability of v_i. A conditional independence assumption is made for such networks: $P(v_i|\pi_i, v_j) = P(v_i|\pi_i)$ where v_j is any variable other than v_i itself and its descendants‡. [2] This assumption is much more reasonable than those made in early Bayesian AI systems[19,22], and it allows one to recover the joint probability distribution of all variables from the given CPD by the simple equation

$$P(v_1, v_2, \dots, v_n) = \prod_{i=1}^{n} P(v_i|\pi_i). \tag{1}$$

In this paper, the node subscripts are ordered as follows. The first m nodes (v_1 through v_m) are the m roots of the network; for any pair v_i and v_j of the remaining $n - m$ non-root nodes, $i < j$ only if v_i is not a descendent of v_j in the network. We call such ordering a *causal precedence order* of the nodes. We use $\{v_i,\ i < j\}$ to denote the set of all nodes that precede the non-root node v_j (including all nodes in π_j). With such an ordering, the joint probability distribution is then given by

$$P(v_1, v_2, \dots, v_n) = \prod_{i=1}^{m} p_i \cdot \prod_{j=m+1}^{n} P(v_j|v_k,\ k < j). \tag{2}$$

The otherwise unrestricted CPD makes this model very general in representing the relationship between a node and its direct causes (parents). However, such generality may require excessive

[2]‡ One can see from this assumption that all root nodes are independent of each other.

number of probabilities be available in *CPD* because the number of conditional probabilities for each node is exponential to the number of its parents. This problem becomes more serious for networks of high connectivities[15]. To reduce the number of conditional probabilities required for a network, a class of simplified causal networks, known as networks of disjunctive interaction or noisy-or gates, has been developed in the past[4,22,20]. In such networks, instead of conditional probability distributions, each causal link is associated with a *single* probability, called *causal strength*, which measures the strength the parent node is to cause the child node. The number of probabilities needed to represent the causal relations among the nodes of a network is thus reduced to the number of its causal links. Another advantage of noisy-or networks is that the one-to-one correspondence between causal links and causal strengths makes them more natural to humans than the conditional probability distribution $P(v_i|\pi_i)$ of general belief networks. This feature of noisy-or networks is particularly valuable when domain experts are involved in acquisition and assessment of causal knowledge for belief network constructions.

Adopting the formulation of noisy-or network from Peng and Reggia[22], the causal strength c_{ij} for link $< v_i, v_j >$ is defined in our method as

$$c_{ij} = P(v_i = 1 \textit{ actually and directly causes } v_j = 1 | v_i = 1), \tag{3}$$

representing the strength of the causal association between v_i and v_j. We consider "$v_i = 1$ actually and directly causes $v_j = 1$" a special event, called *causation* from v_i to v_j, denoted "κ_{ij}"§.[3] We also use p_i to abbreviate $P(v_i = 1)$. Then from the definition of c_{ij} above, we have $c_{ij} \cdot p_i = P(\kappa_{ij} = 1)$. When there does not exit a causal link from v_i to v_j, then $c_{ij} = 0$. In other words, v_i is a direct cause of v_j and thus there is a link from v_i to v_j *if and only if* $c_{ij} > 0$. Therefore, the network structure (i.e., the *DAG*) can be completely depicted by the causal strengths c_{ij} for all $j > m, i < j$. Consequently, the objective of learning a noisy-or network is to obtain from the case data

1) p_i for $i = 1, \cdots, m$, and

2) c_{ij} for $j = m+1, \cdots, n$ and $i < j$.

[3]§ Others[20,18] attach the quantity "$1 - c_{ij}$" to the causal link from v_i to v_j.

The structure of the learned network can then be obtained by keeping only those links whose learned causal strengths are greater than zero (or a very small positive number chosen as the cut off value).

Please note the distinction between the causation event "κ_{ij}" and the more familiar conjunctive event "$v_i \wedge v_j$". The latter represents an association of co-occurrence while the former an association of causality. Conjunctive events can be observed in statistical case data while causations cannot and their occurrences must be inferred or guessed. For example, consider the small network of four nodes in Figure 1. Suppose v_1, v_2, v_3 and v_4 all have value 1 in a particular case. We may conclude that both causations $\kappa_{1,2}$ and $\kappa_{1,3}$ are true in this case if we assume that no non-root variable can be true without being caused by some of its parent(s). However, we cannot conclude with certainty that $\kappa_{2,4}$ is true because v_4 may actually be caused by v_3, not by v_2. For this reason, we consider causations implicit or *hidden*: they are hidden from observed case data.

Figure 1 about here

Furthermore, learning must distinguish the genuine direct causal associations from other associations. Two well-known types of the latter associations are *spurious associations* or co-variances (stemmed, for example, from the correlation between two nodes sharing a common ancestor) and *indirect causations* (e.g., one node is an ancestor but not a parent of another node). For the network in Figure 1, a spurious association exists between nodes v_2 to v_3 due to their common parent v_1, and an indirect causation exists from v_1 to v_4 which can be realized either through v_2 or v_3, or both. Here we must avoid to mistakenly establish causal links between v_2 and v_3, and from v_1 to v_4.

Before leaving this brief review of the basics of noisy-or belief networks, we note that the conditional probability distributions for non-root nodes can be easily computed from the causal strengths. Specifically, for non-root nodes v_j,

$$P(v_j = 1|\pi_j) = 1 - \prod_{v_k \in \pi_j} (1 - c_{kj} v_k) = 1 - \prod_{k<j} (1 - c_{kj} v_k) \tag{4}$$

Here the first equality is from the operation of "noisy-or gate"[20,22], the second equality from that $c_{kj} = 0$ for those v_k's which precede v_j but do not have direct causal links to v_j.

3 The Two Learning Rules

In this section we develop the two learning rules. The learning process takes as inputs the case descriptions which are complete instantiations of all variables v_1 through v_n. Specifically, we denote the t^{th} case input by a binary vector $V(t) = (v_1(t) \ldots, v_n(t))$. We omit t when doing so does not cause confusion. Inputs can be either continuously generated from the environment or fetched (possibly repeatedly) from a fixed database of existing cases. The fundamental assumption underlies this learning method is that the process that generates the case inputs can be closely modeled by a noisy-or network. We call such an underlying network *the target network* and use p_i to denote the prior probabilities of its nodes and c_{ij} the causal strengths of its links. The learning network, also of n nodes, uses primarily two classes of learning variables. They are

1) $x_i, i = 1, \cdots, n$, to be trained to p_i's; and

2) $w_{ij}, j = m+1, \cdots, n, i < j$, to be trained to $P(\kappa_{ij} = 1)$.

Then the third class of learning variables y_{ij}, the learned causal strengths, can be computed by w_{ij}/x_i. When x_i and w_{ij} approach p_i and $P(\kappa_{ij} = 1)$, respectively, y_{ij} approaches c_{ij}. We further assume that a causal precedence order of all of the n nodes is known prior to learning. Node orderings similar to the causal precedence order here is also assumed by Cooper and Herskovits[6] and by Neal[18]¶. Such ordering prevents the formation of causal cycles. Moreover, it restricts the set of all possible parents of a non-root node v_j to $\{v_i, i < j\}$, the set of all nodes preceding v_j. [4] Learning will then determine the genuine parents of v_j only from this restricted set of possibilities. For example, by the node ordering given in the network in Figure 1 (assuming v_1 is the only root), we only need to identify the genuine causal links from the total of 6, not of 12, possibilities.

Next we present and analyze the two learning rules for x_i and w_{ij}, respectively.

[4]¶ Neal suggests[18] a way to learn to distinguish roots from non-root nodes with a node ordering which does not explicitly specifying the roots. This is done by introducing a pseudo node v_0 which precedes all other nodes in the given ordering and has the prior probability of 1. It can be shown that, when sufficient learning inputs are applied, the learned causal strength from v_0 to any root node is the root's prior probability (> 0), and to any non-root node is zero. This is certainly applicable to our learning method, although we do not incorporate it into this paper.

3.1 Learning Probabilities of Explicit Events

The first learning rule can be used to learn the probability of any explicit (i.e., observable) random event. Let e denote such an event, and let $e(t) = 1$ if the t^{th} input confirms the occurrence of the event, and $e(t) = 0$, otherwise. Let variable $x(t)$ denote the learned probability of e after responding to $e(t)$ in the t^{th} input. Then we define

$$\text{Rule 1:} \quad x(t) = x(t-1) + \alpha(e(t) - x(t-1)),\ t > 0, \tag{5}$$

where initially $x(0) \in (0,1)$, and $0 < \alpha \ll 1$ is the learning rate. It is clear that a positive external stimulus ($e(t) = 1$) will cause $x(t)$ to increase, while a negative stimulus ($e(t) = 0$) will cause it to decrease. Moreover, since $x(0) \in (0,1)$ initially, $x(t) \in (0,1)$ for all $t > 0$.

To analyze this learning rule, rewrite Eq. (5) as

$$x(t) = (1-\alpha)^t x(0) + \alpha \sum_{k=0}^{t-1} (1-\alpha)^k e(t-k). \tag{6}$$

Note that in the summation in Eq. (6), each $e(t-k)$ is weighted by a factor of $(1-\alpha)^k$. This makes this equation identical to the approximate formula for "exponentially weighted moving average" (EWMA) method in statistical forecasting for the mean value of a sequence of binary samples[7]. As with EWMA, if e occurs independently in the inputs with constant probability P_e, then we can have the following. The expected value of $x(t)$ is

$$\begin{aligned} E[x(t)] &= (1-\alpha)^t x(0) + \alpha \sum_{k=0}^{t-1} (1-\alpha)^k P_e \\ &= (1-\alpha)^t x(0) + P_e(1-(1-\alpha)^t). \end{aligned} \tag{7}$$

Thus, $E[x(t)] \to P_e$ when $t \to \infty$. The variance of $x(t)$ is

$$\begin{aligned} Var[x(t)] &= \alpha^2 \sum_{k=0}^{t-1} (1-\alpha)^{2k} P_e(1-P_e) \\ &= \frac{\alpha}{2-\alpha}(1-(1-\alpha)^{2t}) P_e(1-P_e). \end{aligned} \tag{8}$$

Thus, $Var[x(t)] \to \alpha P_e(1-P_e)/(2-\alpha)$ when $t \to \infty$.

Combining the above two results, if t is sufficiently large, then $x(t)$ can be trained, regardless of the initial $x(0)$, to oscillate around P_e with standard deviation $\sqrt{Var[x(t)]}$ which is small when

α is small. It is in this sense that we say $x(t)$ can be trained to P_e. Note that $Var[x(t)]$ cannot be reduced to zero by prolonging the learning process if α is constant. However, if α is allowed to slowly decrease toward zero, $Var[x(t)]$ approaches zero and $x(t)$ converges to P_e when $t \to \infty$. This learning rule is in favor of later occurrences of e when a constant α is used, because when t is large, terms of early inputs (i.e., those with large k in their weights $(1-\alpha)^k$ in Eq. (6)) become negligible. In other words, as learning progresses, occurrences of e in early inputs are gradually forgotten, and what is learned is the probability of the *recent* rather than the entire history of the event. This phenomenon, called *limited memory span* in neural network literature[14], reflects the adaptive nature of the learning method and explains the non-zero variance of Eq. (8). In situations where event probability distributions undergo slow changes, this property may make $x(t)$ well adapted to such changes. Also, if the initial $x(0)$ is set to an estimate of P_e obtained from other sources, Rule 1 can be used to refine $x(t)$ according to inputs from a particular environment.

The Hebbian rule of neural networks requires an increase of the synaptic weight between two neurons whenever both neurons are activated. This rule has several mathematical forms, one of which is given below[13,14].

$$\Delta s_{ij}(t) = \alpha(a_i(t)a_j(t) - s_{ij}(t)) \tag{9}$$

where s_{ij} is the synaptic weight from node i to j, and a_i and a_j are their activations[14]. Rule 1 of Eq. (5) can be seen as a special case of the Hebbian rule. If a_j takes fixed value 1 and $a_i = e$, then Eq. (9) becomes $\Delta s_{ij}(t) = \alpha(e(t) - s_{ij}(t))$, which is the same as Eq. (5). If $a_i = e_1$ and $a_j = e_2$, then Eq. (9) becomes $\Delta s_{ij}(t) = \alpha(e_1(t) \wedge e_2(t) - s_{ij}(t))$, which is again the same as Eq. (5) with e now being event "$e_1 \wedge e_2$", and s_{ij} being trained to the joint probability $P(e_1 = 1 \wedge e_2 = 1)$.

It is straightforward to use Rule 1 to learn p_i, the prior probability of v_i of a belief network. Let x_i replace x and v_i replace e in Eq. (5). Then x_i will be changed by a small amount with each input. The direction of the change depends on whether v_i has value 1 or 0 in a given input. Eventually x_i will be trained to oscillate around p_i with a small variance if sufficient number of inputs are applied. Moreover, Rule 1, if needed, can be used to learn not only the marginal probability of v_i, but also the joint probability of v_i and v_j, and in general the joint probability of any conjunctive event whose occurrences can be confirmed by the case inputs.

3.2 Learning Probabilities of Hidden Causation Events

Learning causal strengths c_{ij} is more important than learning prior and joint probabilities because, as stated in Section 2, it is these causal strengths that define the causal relationships between nodes, and thus the structure of a network. Analogous to Eq. (5) of Rule 1, if the occurrences of κ_{ij} can be determined from case inputs, then the rule that trains w_{ij} to $P(\kappa_{ij} = 1)$ should be

$$w_{ij}(t) = w_{ij}(t-1) + \alpha(\kappa_{ij}(t) - w_{ij}(t-1)).$$

The difficulty of such a direct application of Rule 1 is that κ_{ij} are in general *not* observable in the input. Next, we show how Rule 1 can be extended to overcome this difficulty.

Since $\kappa_{ij} = 1$ implies both $v_i = 1$ and $v_j = 1$, then $\kappa_{ij} = 0$ if the current input has $v_i = 0$ or $v_j = 0$, and thus w_{ij} should decrease in the same way as given by Rule 1. On the other hand, when both v_i and v_j have value 1 in an input, κ_{ij} may not necessarily be true if there are some other nodes with $v_k = 1$ and $k < j$ also in the same input and thus can cause v_j to occur. In other words, the occurrence of κ_{ij}, and consequently the way in which w_{ij} is to be adjusted, is uncertain in such situations. This uncertainty can be measured by $P(\kappa_{ij} = 1 | v_j = 1, v_i = 1, v_k, k < j, k \neq i)$, the probability of κ_{ij} being true, given the particular input in which both $v_j = 1$ and $v_i = 1$. If all causal strengths of the belief network is known, then we have (see Appendix for proof)

$$P(\kappa_{ij} = 1 | v_j = 1, v_i = 1, v_k, k < j, k \neq i) = \frac{c_{ij}}{1 - \prod_{k<j}(1 - c_{kj} v_k)}. \tag{10}$$

This, as the probability of $\kappa_{ij} = 1$, is the probability that w_{ij} will increase for an input containing $v_i = 1$ and $v_j = 1$. However, since the actual causal strengths c_{kj} are not available during learning, this probability is not available. Instead, we use the current y_{kj} as approximation of c_{kj} and define the following quantity as an estimate of this probability.

$$q_{ij}(t) = \frac{y_{ij}(t-1)}{1 - \prod_{k<j}(1 - y_{kj}(t-1) v_k(t))}. \tag{11}$$

The second learning rule is then given in the following.

$$\text{Rule 2:} \quad w_{ij}(t) = w_{ij}(t-1) + \alpha(v_i(t)v_j(t)q_{ij}(t) - w_{ij}(t-1)). \tag{12}$$

It can easily be seen from Eq. (12) that, if either v_i or v_j equals zero in the input, then w_{ij} decreases by a factor of $(1-\alpha)$, as is given by Rule 1. Otherwise (both v_i and v_j equal one), w_{ij} increases by an amount depending on q_{ij}. If v_i is the only potential parent of v_j with value 1, then $q_{ij} = 1$ and Rule 2 is again reduced to Rule 1 with w_{ij} increasing by $\alpha(1 - w_{ij})$. When some $v_k = 1, k < j, k \neq i$ also in the input, the increase of w_{ij} will depend on the difference between q_{ij} and the current w_{ij}, reflecting the uncertain nature of the update for such situations.

Note that in updating w_{kj}, all $v_k = 1, k < j$ have the same denominator in their respective q_{kj}. These potential parents v_k of v_j are thus grouped together. They compete with each other in increasing their w_{kj} with the respective y_{kj} as the competition factors, normalized by the common denominator of q_{kj}. As a result, updating w_{ij} depends not only on v_i and v_j in the input and the current values of x_i and w_{ij}, but also on the values of v_k's and the current values of y_{kj}'s for all $k < j$. This competition-based update of Rule 2 is crucial, as will be seen shortly, for the effectiveness and efficiency of this method. Also note that w_{ij} is independent of all other y_{kl}'s with $l \neq j$.

In summary, the core of the learning algorithm, involving the update of all learning variables with one case input $V(t)$, consists of the following three simple steps:

1. update $q_{ij}, j = m+1, \cdots, n, i < j$ by Eq. (11);
2. update $x_i, 1 = 1, \cdots, n$ by Eq. (5);
3. update $w_{ij}, j = m+1, \cdots, n, i < j$ by Eq. (12).

Note here when use Eq. (11) in Step 1, y_{ij} are computed by w_{ij}/x_i.

3.3 Convergence Analysis of the Learning Rules

When the two learning rules are applied to an infinite sequence of learning inputs generated from the environment, due to the stochastic nature of the inputs, all learning variables x_i and w_{ij}, except those w_{ij} with the corresponding $c_{ij} = 0$, will never converge to fixed values. As shown by our experiments in the next section, these variables become oscillating around their respective target values after large number of inputs are applied. Analysis of such stochastic dynamic systems is very difficult. In this section, instead, the learning inputs are drawn from a fixed database of cases, and learning is performed in the so-called *epoch mode*. Epoch mode learning is widely used in many neural network learning methods such as backpropagation and Boltzmann machine networks where in each epoch every learning input in the database is examined, and the desired change of each learning variable based on the input is computed. The actual changes of the learning variables then take place at the end of each epoch based on the sum of the desired changes computed from all individual case inputs. Accordingly, the two learning rules now take slightly different forms as follows.

Let Γ denote the fixed database of N learning cases and t be the number of epochs.

$$\text{Rule } 1': \quad \Delta x_i(t) = \frac{\alpha}{N} \sum_{V^r \in \Gamma} (v_i^r - x_i(t));$$

$$\text{Rule } 2': \quad \Delta w_{ij}(t) = \frac{\alpha}{N} \sum_{V^r \in \Gamma} (v_i^r v_j^r q_{ij}^r - w_{ij}(t))$$

where v_i^r denotes the i^{th} variable in the r^{th} input in database Γ. Re-writing these two rules as differential equations, we have

$$\dot{x}_i(t) = \frac{\alpha}{N} \sum_{V^r \in \Gamma} (v_i^r - x_i(t)), \tag{13}$$

$$\dot{w}_{ij}(t) = \frac{\alpha}{N} \sum_{V^r \in \Gamma} (v_i^r v_j^r q_{ij}^r - w_{ij}(t)). \tag{14}$$

Since variable x_i is constant within an epoch, Eq. (13) can be re-written as $\dot{x}_i(t) = \alpha(p_i - x_i(t))$, where $p_i = \frac{1}{N} \sum_{V^r \in \Gamma} v_i^r$ is the marginal probability of $v_i = 1$ in Γ. The analytic solution to Eq. (13) can then easily be found to be

$$x_i(t) = p_i + (x_i(0) - p_i)e^{-\alpha t}. \tag{15}$$

This solution leads to the following theorem, whose proof is trivial and thus omitted.

Theorem 1: When Rule $1'$ (Eq. (13)) is used with $x_i(0) \in (0,1)$, then $x_i(t) \in (0,1)$ $\forall t \geq 0$ and $x_i(t) \to p_i$ when $t \to \infty$.

Eq. (14) of Rule $2'$ defines a dynamic system in which w_{ij}'s of potential parents of node v_j are interacting with each other via the common denominator of their respective q_{ij}'s. Although this system is no longer stochastic, analytical solution to Eq. (14) is still very difficult to obtain. In the following, we analyze the convergence and other properties of this system using a *Liapunov function* (also known as an *energy function* in the neural network literature[11], due to its analogy to energy in thermodynamics). Briefly, a Liapunov function, defined over the state space of a dynamic system, is a monotonically increasing function over time with a finite upper bound. If such a function can be found for the system, then it will always increase until the system eventually settles down into a stable equilibrium state where the function reaches a local maximum and its time derivative (asymptotically) becomes zero. That is, the system is guaranteed to converge.

Let $\widetilde{V} = (v_1 \ldots, v_n)$ denote a state of the learning network. The function, which will be shown to be a Liapunov function of the system of Eq. (14), is defined below.

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

In statistics, this function measures the likelihood that the learning network is to produce the case data in Γ, and thus measures the similarity between the joint probability distributions of the two. By Eq. (2) in Section 2,

$$P(\widetilde{V} = V^r) = \prod_{i=1}^{m} P(\tilde{v}_i = v_i^r) \prod_{j=m+1}^{n} P(\tilde{v}_j = v_j^r | v_k^r, k < j).$$

Function L can then be written as

$$L = \sum_{V^r \in \Gamma} [\sum_{i=1}^{m} \ln P(\tilde{v}_i = v_i^r) + \sum_{j=m+1}^{n} \ln P(\tilde{v}_j = v_j^r | v_k^r, k < j)], \tag{16}$$

where $P(\tilde{v}_i = v_i^r = 1) = x_i$, and from Eq. (4)

$$P(\tilde{v}_j = v_j^r | v_k^r, k < j) = \begin{cases} 1 - \prod_{k<j}(1 - y_{kj} v_k^r) & \text{if } v_j^r = 1; \\ \prod_{k<j}(1 - y_{kj} v_k^r) & \text{if } v_j^r = 0. \end{cases} \tag{17}$$

To show L is a Liapunov function, we must show that 1) L has an upper bound and 2) L is monotonically increasing with time. The first requirement is easy to confirm because both $P(\tilde{v}_i = v_i^r)$ and $P(\tilde{v}_j = v_j^r | v_k^r, k < j)$ are between 0 and 1. This is put in the lemma below.

Theorem 2: $L \leq 0$.

To establish the second requirement, we first analyze $\dot{w}_{ij}$ and $\partial L/\partial w_{ij}$ in the next lemma. (See Appendix for proof.)

Lemma: Let $\Gamma_1 \subseteq \Gamma$ be the set of all case inputs each of which contains $v_i = 1$ and $v_j = 1$ for the given $i < j$. Then,

$$\dot{w}_{ij} = \frac{\alpha}{N} \cdot y_{ij} \cdot [\sum_{V^r \in \Gamma_1} \frac{1}{1 - \prod_{k<j}(1 - y_{kj}v_k^r)} - x_i N], \tag{18}$$

and

$$\frac{\partial L}{\partial w_{ij}} = \frac{1}{(1 - y_{ij})x_i}[\sum_{V^r \in \Gamma_1} \frac{1}{1 - \prod_{k<j}(1 - y_{kj}v_k^r)} - p_i N]. \tag{19}$$

Now we show the main result that L is monotonically increasing after some point of time.

Theorem 3: There exist $t_0 > 0$ such that when $t > t_0$, $\dot{L}(t) \geq 0$.

Proof: Let $A_j = \sum_{V^r \in \Gamma_1}(1 - \prod_{k<j}(1 - y_{kj}v_k^r))^{-1}$ and $c = N(x_i(0) - p_i)$. By Eqs. (15) and (18),

$$\dot{w}_{ij} = \frac{\alpha}{N} \cdot y_{ij} \cdot [(A_j - p_i N) - ce^{-\alpha t}].$$

Then by the above equation, Eq. (19), and the chain rule of derivatives

$$\begin{aligned} \dot{L}(t) &= \sum_{i,j} \frac{\partial L}{\partial w_{ij}} \cdot \dot{w}_{ij} \\ &= \sum_{i,j} \frac{y_{ij}}{(1 - y_{ij})x_i} \cdot \frac{\alpha}{N} \cdot [A_j - p_i N] \cdot [(A_j - p_i N) - ce^{-\alpha t}]. \end{aligned}$$

Since $0 < y, x < 1$, if there exists some $t_0 \geq 0$ such that $|A_j(t) - p_i N| \geq |ce^{-\alpha t}|$ for $t \geq t_0$ for all j, then $\dot{L}(t) \geq 0$ for $t \geq t_0$. It is obvious that such t_o exists if $A_j(t) - p_i N$ does not converge to zero for any j. Otherwise, since the convergence of A_j depends on the convergence of w, which, as can be seen from Eq. (14), has a convergence rate slower than that of $ce^{-\alpha t}$, such t_0 also exists. $\square$

When the number of learning epochs is sufficiently large, not only $L(t)$ is guaranteed to converge (asymptotically) to a stable state of a local maximum value, but also the learned network has its $P(\tilde{V})$ distribution very close to the corresponding distribution in database Γ. To conclude the analysis, we note that the underlying target network from which vectors in Γ are generated is one such state and thus an attractor of the dynamic system. This is because the target network must be one of the networks that have the maximum likelihood (i.e., L value) to generate Γ. We also note that our learning rules, when working in epoch mode, only guarantee to find a belief network with a locally maximum value of L. However, when working in sequential mode, due to the limited memory span of the learning rules, the learning network may be able to forget, and thus escape from, the local maxima and continue to look for the global maximum by subsequent input presentations.

4 Experimental Results

Two computer experiments are conducted to test the validity of the learning method. The first experiment uses a randomly generated bipartite noisy-or network taken from Peng and Reggia[22]. As illustrated in Figure 2, this network consists of two layers of nodes. The 10 nodes at the top layer are roots and the 10 nodes at the bottom layer are non-root nodes. The two layers of nodes can be considered as causes and effects, respectively, in a causal model, and as disorders and manifestations in a diagnostic setting. There are totally 35 causal links in this network, all of which are from the nodes at the top layer to nodes at the bottom layer. Each root has a prior probability p_i, ranging from 0.014 to 0.098; and each link has a causal strength c_{ij}, ranging from 0.11 to 0.96. This network serves two purposes in the experiment: as a learning input generator to randomly generate input vectors V; and as a target network to compare with the learned results.

Figure 2 about here

The learning network is also a 10 by 10 bipartite network. Since a bipartite network does not allow intra-layer causal links, the set of all cause nodes (nodes 1 to 10) is the set of all potential parents for each of the 10 effect nodes (nodes 11 to 20) in the learning network. The causal precedence order, mainly used to determine the potential parents for each non-root node, is then not needed here. Initially, x_i is set to 0.0001 for $i = 1$ to 10, and w_{ij} to 0.00001 for $i = 1$ to 10, $j = 11$ to 20.

There are thus 110 probabilities to be learned in this experiment, 10 x_i's for the 10 roots and 100 w_{ij}'s for each i and j. Among the 100 w_{ij}'s, all but those 35 which correspond to causal links should be learned to approach zero. The learning rate α is chosen to be very small (0.001).

Upon receiving an input, the learning network adjusts its x_i and w_{ij} according to Rules 1 and 2, respectively. For a more meaningful evaluation of learning errors in comparison to the target values, we divide the learning variables x's and y's into three groups: Group 1 for all y_{ij}'s whose corresponding $c_{ij} > 0$; Group 2 for the rest of y_{ij}'s ($c_{ij} = 0$); and Group 3 for all x_i's. After about 2,000 input presentations, y's in Group 2 are all reduced to less than 0.0082 while those in Group 1 are greater than 0.099. The difference of more than an order of magnitude between these two groups of variables is sufficient, in our view, to determine if a genuine direct causal association exists between a root and a non-root node. We thus claim the structure of the network is successfully learned. Continuing the experiment with additional inputs, all y's in Group 2 continue to decline toward zero while other y's (in Group 1) and x's (in Group 3) become oscillating with small fluctuations around their respective target values. This trend continues until the experiment ends at 6,000 input presentations. By then, the maximum value of all y's in Group 2 is reduced to 0.00002, the minimum value of all y's in Group 1 is greater than 0.09.

To quantitatively measure the overall learning quality, for each of the three groups of variables we compute the average absolute errors (AAE) at different time points. The error curves for the three groups are plotted in Figure 3. These curves clearly demonstrates that learning progresses as expected. The non-zero AAE's for Groups 1 and 3 reflect the non-zero variances of these variables.

Figure 3 about here

The second experiment uses a substantially larger and more complex network of 37 nodes (12, roots, 8 leaves and 17 intermediate nodes) and 46 links. This network, given in Figure 4, has the structure of the "Alarm belief network", a model for potential anesthesia problems in the operation room, used in Cooper and Herskovits[6]. Since our learning rules work only with binary variables,

variables are treated as binary in the experiment, although they are multi-valued in the original Alarm network. The prior probabilities of roots and causal strengths of links of this network are generated randomly by us for the experiment. They range from 0.15 to 0.89, and from 0.11 to 0.92, respectively. The causal precedence order among non-root nodes used in the experiment is also taken from Cooper and Herskovits[6].

Figure 4 about here

The learning network is again initialized with $x_i = 0.0001$ for every node v_i, and $w_{ij} = 0.00001$ for every potential causal link in the network. The learning rate α affects both the quality and efficiency of learning. Large α leads to a fast convergence but may cause large variances in the learned variables (see Eq (8)). This may result in mis-classification of causal links from time to time. As a trade-off, instead of using a fixed α, we employ in this experiment a more sophisticated schedule to gradually reduce the learning rate from a relatively large initial value. After 5,000 input presentations (with $\alpha = 0.1$ for the first 2000 input and $\alpha = 0.01$ for the next 3000 inputs), the causal structure is learned with the maximum value of y's in Group 2 equal to 0.0027 and the minimum value of y's in Group 1 equal to 0.0758. Additional 5000 inputs with $\alpha = 0.001$ bring all y's in Group 2 to even closer to zero and all y's in Group 1 and all x's in Group 3 to oscillate around their respective target values. The average absolute errors for the three groups of variables are plotted in Figure 5. The curves have similar shapes as those in Figure 3 for the bipartite network after sufficiently many inputs.

Figure 5 about here

The success of the second experiment, although with a longer learning time, is quite remarkable not only because of the size of the network (47 out of the total of 600 y's must be distinguished as representing genuine causations) but also the complexity of its structure (e.g., some indirect causation goes as deep as 9 levels).

5 Comparisons With Some Related Work

In this section, we compare our work with two other methods for learning belief networks from case data. The first of the two is Neal's work[18] which is probably the closest to our method among all existing methods. The other is the K2 algorithm by Cooper and Herskovits[6] which is one of the first general learning methods for belief networks. Neal takes the approach of Boltzmann machine of neural networks to learning two types of belief networks, namely *sigmoid* and *noisy-or* networks. One distinct feature of this method is that it allows the learning network to contain a number of hidden nodes (nodes without corresponding variables in the case input vectors) for a better fit of the data. (One drawback of allowing hidden nodes is that these nodes and their related learned association strengths may not have clear semantics.) This approach applies gradient ascent method to maximize the log-likelihood function L below

$$L = \ln \prod_{V \in \Gamma} P(\tilde{V} = V), \tag{20}$$

which, when no hidden node is involved, becomes the same L that we identified in Subsection 3.3 as the Liapunov function for our method. Learning takes place by updating the inter-node connection strength s_{ij} where s_{ij} relates to our causal strength as follows[18].

$$exp(-s_{ij}) = 1 - y_{ij}. \tag{21}$$

For a fair comparison, in what follows we consider only noisy-or networks without hidden nodes for Neal's method. When working in sequential mode, the update rule for s_{ij} can be written as

$$\Delta s_{ij} = \begin{cases} -v_i + v_i/(1 - exp(-\sum_{k<j}(v_k s_{kj}))) & \text{if } v_j = 1; \\ -v_i & \text{if } v_j = 0. \end{cases} \tag{22}$$

Several points of similarity between Neal's method and ours can be noted immediately. First of all, both methods are trying to adjust the learning network to maximize the similarity of the probability distributions between the network and the database according to the same likelihood function L. Secondly, note from Eq. (21) that a larger s_{ij} corresponds to a larger y_{ij}, and that s_{ij}

moves from zero to infinity when y_{ij} moves from zero to one. Similar to our method, when any s_{ij} is trained to approach zero, then we can consider that there is no direct causal relation from node v_i to node v_j and thus no causal link from v_i to v_j. Thirdly, note from Eq. (22) that s_{ij} changes its value with an input only if the parent node $v_i = 1$ in that input, and the direction of the change depends on whether the child node v_j is 1 or 0 in the input. This is qualitatively the same as for y_{ij} in our method. Moreover, when both v_i and v_j are 1 in the input, the amount of increase in s_{ij} depends on the quantity $1 - exp(-\sum_{k<j}(v_k s_{kj})$, which, by Eq. (21), equals $1 - \prod_{k<j}(1 - y_{kj}v_k)$, the denominator of q_{ij} in our second learning rule. Nevertheless, there is a significant difference between the two methods. In our method, as discussed in Subsection 3.2, all potential parent nodes v_k of v_j with $v_k = 1$ in the input compete with each other and the actual amount of increase in w_{kj} is proportional to the current value of y_{kj}. However, there is no competition between potential parent nodes in Neal's method. For $v_j = 1$ in an input, all its potential parents v_k with value 1 in the input will have their s_{kj} increased by the same amount.

To empirically compare Neal's method with ours, we tested Neal's method (using Eq. (22) as the learning rule) on both the bipartite network in Figure 2 and the Alarm network in Figure 4. In both experiments only learning results for s_{ij}'s are collected and then converted from s_{ij} to y_{ij}. In the first experiment, all causal links can be learned by 4,000 inputs. The average absolute errors for the experiment are plotted in Figure 6, where the two curves are for Group 1 variables (i.e., those y_{ij}'s with the targets $c_{ij} > 0$) and Group 2 variables (those y_{ij}'s with the targets $c_{ij} = 0$), respectively.

Figure 6 about here

Comparing Figure 6 with Figure 3, it can be noted that the two methods have their AAE curves of Group 2 variables approaching zero with similar speed. However, Neal's method has much larger average errors for Group 1 variables than our method, indicating that many more inputs are needed in order to bring these probabilities close to their target values. In the second experiment, we have tried many different schedules for the learning rate α, each with a substantial number of inputs.

Although Group 1 variables can be properly trained to approach their target values, none of these schedules is able to produce a clean cut learning result in which all variables of Group 2 are below a given cut-off value for genuine direct causal associations (set to 0.01 for all experiments). In other words, we are unable to obtain a learning result without including some extraneous causal links. The best result is obtained by using the following schedule: α set to 0.01 for the first 5,000 inputs, 0.001 for the following 10,000 inputs, and 0.0001 for the rest of the inputs. The experiment stops after half a million input presentations. The AAE curves are plotted in Figure 7.

Figure 7 about here

Comparing Figure 7 with Figure 5, it is clear that Neal's method is substantially slower than our method in learning the complex Alarm network. Moreover, we have observed that, in contrast to learning using our method, some Group 2 variables using Neal's method in both experiments do not approach zero but oscillate around small positive numbers. From time to time some of their values exceed the cut-off value and thus are misclassified as genuine causal links. We suspect that both the slow learning and the non-convergence of some Group 2 variables by Neal's method are largely due to the lack of competition between potential parents of a given node. Competition in our method not only makes y_{ij} in Group 2 decreasing faster toward zero but also in effect clamps them at close to zero afterward because the very small values of such y_{ij} make them much less competitive in the subsequent updates.

Cooper and Herskovits take an Bayesian approach to learning belief networks. Their method includes a formula which, under certain assumptions, computes the posterior probability of a network structure, given a database of cases. It also includes a search algorithm, called K2, which heuristically finds a very probable network structure for the given data. This method is more general than ours in that it works for both general belief networks and noisy-or networks. Also, this method is more robust in that the learning is not affected by operational parameters such as learning rate used in our method. On the other hand, this method is non-incremental, and thus only works with fixed databases, whereas our method works with both fixed databases and continuous flows of case inputs from the environments. To compare the efficiency and effectiveness of the two methods, we test them over two sets of fixed databases of different sizes generated from the two belief networks in Figures 1 and 4. Since K2 itself only learns network structures, we only test if causal links are properly learned in these experiments. When testing our method, we let the learning process continue for 10,000 input presentations. When a database has fewer than 10,000 case inputs, its entries are repeatedly used in cycles. We choose $\alpha = 0.001$ for the bipartite network. For the Alarm network, α is set to 0.1 for the first 2,000 inputs, 0.01 for the next 3,000 inputs, and 0.001 for the rest 5,000 inputs. The results are given in Tables 1 and 2 where in each entry the number before the slash sign is the result of K2 algorithm and the number after it is the result of our method. In these experiments, the two methods show a same degree of accuracy for the two target networks with different database sizes. However, our method is significantly faster than K2 in all cases.

Table 1 about here

Table 2 about here

6 Conclusions

In this paper we present a simple but effective learning method for noisy-or belief networks. Two learning rules are developed for learning prior probabilities of nodes and causal strengths of links, respectively. Mathematical analysis is conducted, using a Liapunov function, to prove the convergence of the learning process. The validity of this method is confirmed by results of successful experiments with belief networks of both bipartite and general structures. This method is in contrast to most existing methods for learning belief networks[6,9,21,15] in that the learning is incremental and what can be learned are probabilities of recent rather than entire history of the random events. This makes the learned causal network adaptive to slow environmental changes. Although this method is limited to noisy-or belief networks, it is substantially more efficient than some other methods in learning this class of belief networks.

Several issues need further investigation. Using a Liapunov function (Theorems 2 and 3), we proved that the learning process converges when working in epoch mode with a fixed database of cases. We belief, and the experiments also suggest, that when working in sequential mode with case inputs continuously generated from the environment the learning process will also converge, albeit in a stochastic sense very much like Eqs. (7) and (8) for Rule 1. However, the mathematical proof of this conjecture is yet to be established. Also needed to be investigated is the ability of this method to handle case data which are not perfect data with complete instantiations of all variables such as the ones artificially generated from the target networks in our experiments. This may involve the handling of noisy or contaminated data, data containing inference results (thus are not necessarily true reflection of the reality because inferences such as abduction are inherently uncertain), and data containing missing variables. Finally, some practically important issues also need further attention. Examples include the criteria for selecting the learning rate α or its schedule; the criteria for determining when the learning is completed and thus can be terminated; and the number of iterations (repetitions of all case inputs) needed for a successful learning when working with a fixed database. Successfully resolving these and other related issues will hopefully make this method a powerful, effective and practical tool for causal knowledge acquisition.

Appendix

I. *Proof of Equation 10.* From the definition of causation event κ_{ij}, we have

$$P(\kappa_{ij} = 1, v_j = 1, v_i = 1, v_k, k < j, k \neq i) = P(\kappa_{ij} = 1, v_k, k < j, k \neq i)$$

because $\kappa_{ij} = 1$ implies both $v_j = 1$ and $v_i = 1$. Because $v_i = 1$, we also have (see ref. 22)

$$P(\kappa_{ij} = 1, v_k, k < j, k \neq i) = c_{ij} P(v_i = 1, v_k, k < j, k \neq i).$$

Therefore,

$$P(\kappa_{ij} = 1 | v_j = 1, v_i = 1, v_k, k < j, k \neq i)$$

$$
\begin{aligned}
&= c_{ij} P(v_i = 1, v_k, k < j, k \neq i) / P(v_j = 1, v_i = 1, v_k, k < j, k \neq i) \\
&= \frac{c_{ij}}{P(v_j = 1 | v_i = 1, v_k, k < j, k \neq i)} \cdot \frac{P(v_i = 1, v_k, k < j, k \neq i)}{P(v_i = 1, v_k, k < j, k \neq i)} \\
&= \frac{c_{ij}}{1 - \prod_{k<j}(1 - c_{kj} v_k)}.
\end{aligned}
$$

where the last equality comes from Eq. (4). □

II. *Proof of Lemma.* We first prove Eq. (18). Since $v_i^r = v_j^r = 1$ if and only if $V^r \in \Gamma_1$, then

$$
\begin{aligned}
\dot{w}_{ij} &= \frac{\alpha}{N} \sum_{V^r \in \Gamma} (v_i^r v_j^r q_{ij}^r - w_{ij}) \\
&= \frac{\alpha}{N} [\sum_{V^r \in \Gamma_1} q_{ij}^r - \sum_{V^r \in \Gamma} w_{ij}] \\
&= \frac{\alpha}{N} [\sum_{V^r \in \Gamma_1} \frac{y_{ij}}{1 - \prod_{k<j}(1 - y_{kj} v_k^r)} - \sum_{V^r \in \Gamma} w_{ij}] \\
&= \frac{\alpha}{N} \cdot y_{ij} \cdot [\sum_{V^r \in \Gamma_1} \frac{1}{1 - \prod_{k<j}(1 - y_{kj} v_k^r)} - \sum_{V^r \in \Gamma} x_i] \\
&= \frac{\alpha}{N} \cdot y_{ij} \cdot [\sum_{V^r \in \Gamma_1} \frac{1}{1 - \prod_{k<j}(1 - y_{kj} v_k^r)} - x_i N].
\end{aligned}
$$

Here the last equality comes from the fasts that x_i is constant in an epoch and that $|\Gamma| = N$.

To prove Eq. (19), let $\Gamma_2 \subseteq \Gamma$ be the set of V with $v_i = 1$ and $v_j = 0$. From Eq. (16),

$$
\begin{aligned}
\frac{\partial L}{\partial w_{ij}} &= \sum_{V^r \in \Gamma} \sum_{l=m+1}^{n} \frac{\partial}{\partial w_{ij}} \ln P(\tilde{v}_l = v_l^r | v_k^r, k < l) \\
&= \sum_{V^r \in \Gamma_1} \frac{\partial}{\partial w_{ij}} \ln P(\tilde{v}_j = 1 | v_k^r, k < j) + \sum_{V^r \in \Gamma_2} \frac{\partial}{\partial w_{ij}} \ln P(\tilde{v}_j = 0 | v_k^r, k < j).
\end{aligned}
$$

The first equality above comes from that x_i's are not functions of w_{ij} (see Eq. (13)), the second equality from that $P(\tilde{v}_l = v_l^r | v_k^r, k < l)$ does not involve w_{ij} if $v_i^r = 0$ or $l \neq j$ (see Eq. (17)). Therefore, only those $\ln P(\tilde{v}_l = v_l^r | v_k^r, k < l)$ which are in either Γ_1 or Γ_2 and with $l = j$ may have non-zero partial derivatives with respect to w_{ij}.

Then, substituting $P(\tilde{v}_j = v_j^r | v_k^r, k < j)$ using Eq. (17), taking partial derivatives within each summation and performing some algebraic manipulations, we have

$$
\begin{aligned}
\frac{\partial L}{\partial w_{ij}} &= \sum_{V^r \in \Gamma_1} \frac{\partial}{\partial w_{ij}} \ln(1 - \prod_{k<j}(1 - y_{kj} v_k^r)) + \sum_{V^r \in \Gamma_2} \frac{\partial}{\partial w_{ij}} \ln \prod_{k<j}(1 - y_{kj} v_k^r) \\
&= \sum_{V^r \in \Gamma_1} \frac{\prod_{k<j, k \neq i}(1 - y_{kj} v_k^r)}{[1 - \prod_{k<j}(1 - y_{kj} v_k^r)] x_i} - \sum_{V^r \in \Gamma_2} \frac{\prod_{k<j, k \neq i}(1 - y_{kj} v_k^r)}{[\prod_{k<j}(1 - y_{kj} v_k^r)] x_i}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{V^r \in \Gamma_1} \frac{\prod_{k<j}(1-y_{kj}v_k^r)}{[1-\prod_{k<j}(1-y_{kj}v_k^r)](1-y_{ij})x_i} - \sum_{V^r \in \Gamma_2} \frac{1}{(1-y_{ij})x_i} \\
&= \frac{1}{(1-y_{ij})x_i}[\sum_{V^r \in \Gamma_1} \frac{\prod_{k<j}(1-y_{kj}v_k^r)}{1-\prod_{k<j}(1-y_{kj}v_k^r)} - \sum_{V^r \in \Gamma_2} 1] \\
&= \frac{1}{(1-y_{ij})x_i}[\sum_{V^r \in \Gamma_1} (\frac{1}{1-\prod_{k<j}(1-y_{kj}v_k^r)} - 1) - \sum_{V^r \in \Gamma_2} 1] \\
&= \frac{1}{(1-y_{ij})x_i}[\sum_{V^r \in \Gamma_1} \frac{1}{1-\prod_{k<j}(1-y_{kj}v_k^r)} - \sum_{V^r \in \Gamma_1 \cup \Gamma_2} 1] \\
&= \frac{1}{(1-y_{ij})x_i}[\sum_{V^r \in \Gamma_1} \frac{1}{1-\prod_{k<j}(1-y_{kj}v_k^r)} - p_i N].
\end{aligned}
$$

The last equality above comes from that, by the definition of Γ_1 and Γ_2, $V^r \in \Gamma$ is in $\Gamma_1 \cup \Gamma_2$ if and only if $v_i^r = 1$. Therefore, the size of $\Gamma_1 \cup \Gamma_2$ equals the total number of vectors in Γ that have their corresponding $v_i = 1$, which equals $p_i N$. □

References

1. B. Apolloni, G. Avanzini, N. Cesa-Bianchi and G. Ronchini, "Diagnosis of Epilepsy via back-propagation", in *Proc. Intl. Joint Conf. on Neural Networks, II*, 571-574 (1990).

2. R. Blum, *Discovery and Representation of Causal Relationships from a Large Time-Oriented Clinical Database: The RX Project*, New York: Springer-Verlag, 1982.

3. F. Casselman and M. Acres, "DASA/LARS, A Large Diagnostic System Using Neural Networks", in *Proc. Intl. Joint Conf. on Neural Networks, II*, 539-542 (1990).

4. G. Cooper, "A Diagnostic Method that Uses Causal Knowledge and Linear Programming in the Application of Bayes' Formula", *Computer Methods and Programs in Biomedicine*, **22**: 223-237 (1986).

5. G. Cooper, "Current Research Directions in The Development of expert Systems Based on Belief Networks", *Applied Stochastic Models and Data Analysis*, **5**:39-52 (1989).

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

7. W. Gilchrist, *Statistical Forecasting*, New York: John-Wiley & Sons, 1977.

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

9. D. Heckerman 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.), 85-94 (1994).

10. G. Hinton,"Connectionist Learning Procedures", *Artificial Intelligence*, **40**:185-234 (1989).

11. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of the national Academy of Sciences*, **79**, 2554-2558 (1982).

12. O. Jakubowicz and S. Ramanujam, "A Neural Network Model for Fault-Diagnosis of Digital Circuits", in *Proc. Intl. Conf. on Neural Networks, II*, 611-614 (1990).

13. J. Johnson, "Globally Stable Saturable Learning Laws", *Neural Networks*, **4**:47-51 (1991).

14. B. Kosko, *Neural Networks and Fuzzy Systems*, New Jersey: Prentice Hall, 1992.

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

16. W. Levy and B. Burger, "Electrophysiological Observations which Help Describe an Associative Synaptic Modification Rule", in *Proc. IEEE Intl. Conf. on Neural Networks, IV*, 11-15 (1987).

17. B. Mulsant and E. Servan-Schreiber, "A Connectionist Approach to the Diagnosis of Dementia", in *Proc. Sym. on Comp. Applic. in Med. Care*, 245-250 (1988).

18. R Neal, "Connectionist learning of belief Networks", *Artificial Intelligence*, **56**, 71-113 (1992).

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

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

21. 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)

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

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

24. 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 three-layer four-node network.

Figure 2 A 10 $\times$ 10 bipartite network.

Figure 3 Experiment results with the 10 $\times$ 10 network of Figure 2.

Figure 4 The Alarm belief network structure, consisting of 37 nodes and 46 links.

Figure 5 Experiment results with the ALARM network of Figure 4.

Figure 6 Experiment results with the network of Figure 2 using Neal's method.

Figure 7 Experiment results with the network of Figure 4 using Neal's method.

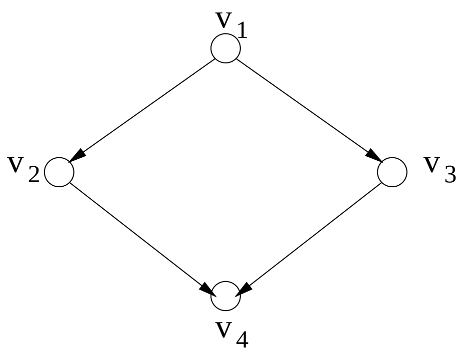


Figure 1.

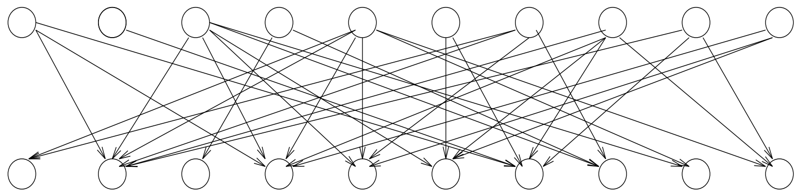
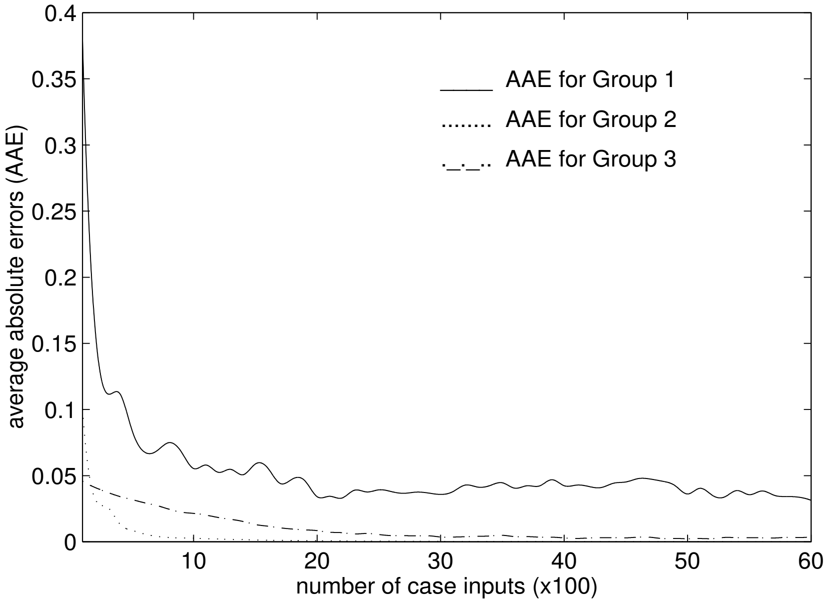


Figure 2.



0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
average absolute errors (AAE)
AAE for Group 1
AAE for Group 2
AAE for Group 3
10
20
30
40
50
60
number of case inputs (x100)

Figure 3.

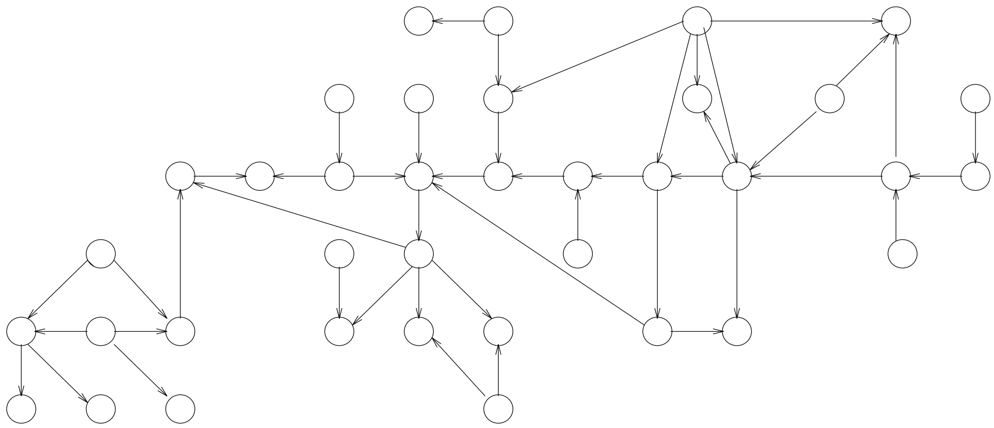


Figure 4.

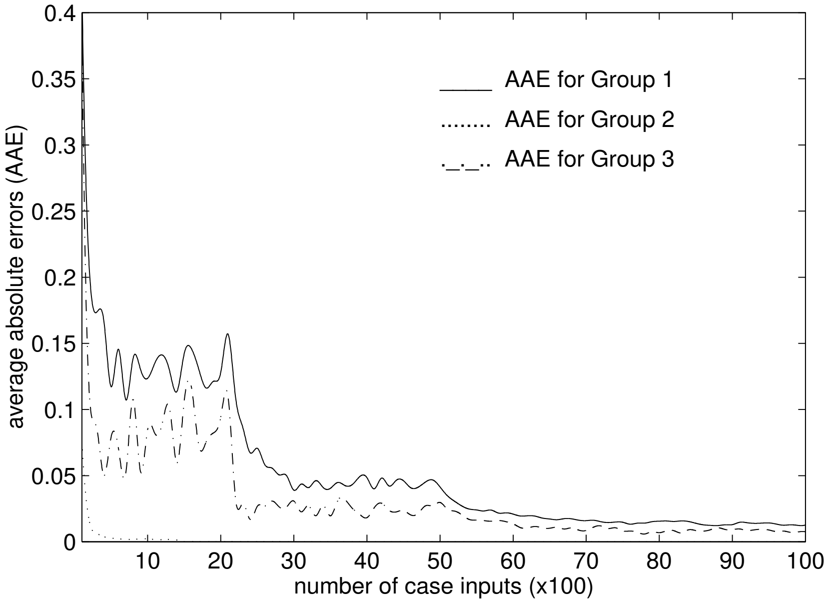
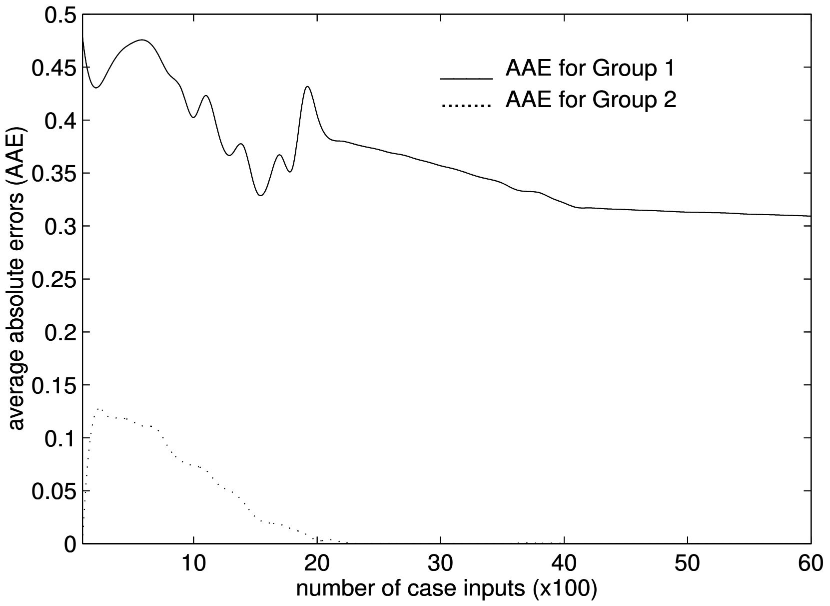

AAE for Group 1
AAE for Group 2
AAE for Group 3
average absolute errors (AAE)
number of case inputs (x100)
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
10
20
30
40
50
60
70
80
90
100

Figure 5.



AAE for Group 1
AAE for Group 2
average absolute errors (AAE)
number of case inputs (x100)
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
10
20
30
40
50
60

Figure 6.

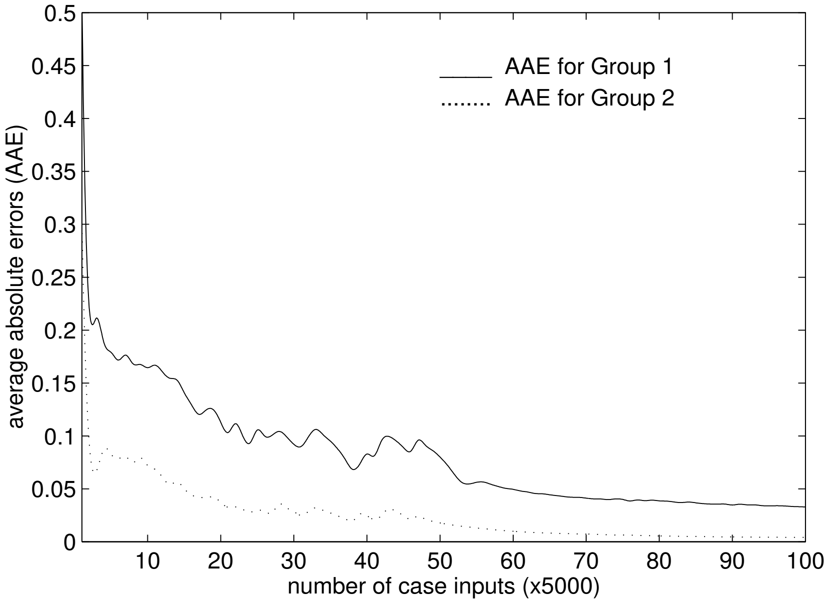

AAE for Group 1
AAE for Group 2
average absolute errors (AAE)
number of case inputs (x5000)
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
10
20
30
40
50
60
70
80
90
100

Figure 7.

Captions for Tables

Table 1 Comparison with K2 algorithm with the network of Figure 2.

Table 2 Comparison with K2 algorithm with the network of Figure 4.

Table 1.

Number of cases	Missing links	Extra links	User Time
500	0/1	6/0	22.72/1.53
1000	0/0	1/0	24.71/1.54
2000	0/0	0/0	27.39/1.54
10000	0/0	0/0	58.09/1.53

Table 2.

Number of cases	Missing links	Extra links	User Time
500	2/0	2/6	63.76/5.91
1000	0/0	1/1	69.62/6.04
2000	0/0	0/0	77.45/5.86
10000	0/0	0/0	161.97/5.83