ELSEVIER Artificial Intelligence 88 (1996) 1–38 Artificial Intelligence

Using hidden nodes in Bayesian networks

Chee-Keong Kwoh, Duncan Fyfe Gillies*

Department of Computing, Imperial College of Science, Technology and Medicine, 180 Queen's Gate, London SW7 2BZ, UK

Received June 1994; revised April 1995

Abstract

In the construction of a Bayesian network, it is always assumed that the variables starting from the same parent are conditionally independent. In practice, this assumption may not hold, and will give rise to incorrect inferences. In cases where some dependency is found between variables, we propose that the creation of a hidden node, which in effect models the dependency, can solve the problem. In order to determine the conditional probability matrices for the hidden node, we use a gradient descent method. The objective function to be minimised is the squared-error between the measured and computed values of the instantiated nodes. Both forward and backward propagation are used to compute the node probabilities. The error gradients can be treated as updating messages and can be propagated in any direction throughout any singly connected network. We used the simplest node-by-node creation approach for parents with more than two children. We tested our approach on two different networks in an endoscope guidance system and, in both cases, demonstrated improved results.

1. Introduction

We will begin with a brief introduction to Bayesian networks, which are also known as belief networks, probabilistic networks, or causal networks. They are graphical structures used for the representation of conditionally independent information which is frequently derived from some subjective knowledge base. Bayesian networks have two components, the dependency graph and the knowledge representation. The underlying structure of the dependency graph, which is sometimes called the inference diagram, fully expresses the dependency or independency of the variables which are represented as the nodes. The knowledge representation component, or knowledge map, stores all the beliefs

* Corresponding author. E-mail: dfg@doc.ic.ac.uk.

PII S0004-3702(95)00119-0

and likelihood information for every possible state of the variables. There are many alternative names given to the states of a variable: Heckerman [9] calls them instances, Neapolitan [20] uses the name propositions, and Richard and Lippmann [26] use classes. In Bayesian networks, all the possible states of the variables are mutually exclusive and exhaustive.

In the closed world definition, if $\boldsymbol{V}$ is the set of variables $\{V_1, V_2, V_3, \ldots\}$ for a model, then

$$P(V) = \prod_{V_i \subset V} P(V_i/\mathrm{pr}(V_i)), \quad V_i, \mathrm{pr}(V_i) \subseteq V \tag{1}$$

where $\mathrm{pr}(V_i)$ is the parent of the variable V_i. In other words, the joint probability of $P(\boldsymbol{V})$ can be expressed as the product of the conditional probabilities of each variable given the state of their parents and $\boldsymbol{V}$ is the set which contains all the possible combinations of V_i. Throughout the text we use boldface to denote a variable, and the corresponding set of probabilities for each state is a vector which will be denoted by an underscore.

Pearl [23] pioneered the use of a graphical representation to express the causality between variables. He uses Bayes' rule:

$$P(A/W) = \frac{P(W/A)P(A)}{P(W)} = \frac{P(W/A)P(A)}{\Sigma_A\, P(W/A)P(A)} = \alpha P(W/A)P(A) \tag{2}$$

to derive the parameters for belief updating.

In the Bayesian networks, belief in a variable is defined as the posterior probability of that variable given all the known evidence, $\boldsymbol{W}$. In a Bayesian network, knowing the states of all connected variables will fully determine the belief in the variable of interest. Knowing all the parents and children of a variable will give a result which is independent of all other variables. In our current application, we have found that the tree structure Bayesian network with only chance nodes is adequate. There are more complex Bayesian networks, and for the determination of causality in them the reader is referred to work by Pearl [23], Geiger et al. [7], and to the work of Lauritzen and Spiegelhalter [17] who use propagation in cliques to cope with loops in Bayesian networks.

Pearl pioneered the use of a directed acyclic graph for representing a Bayesian network. It consists of a set of vertices, $\boldsymbol{V}$, and edges, $\boldsymbol{E}$. The variables are represented by the vertices, and their relationships by directed edges. If $\boldsymbol{A}$ and $\boldsymbol{B}$ are two variables and the parent of $\boldsymbol{B}$ is $\boldsymbol{A}$, then $\boldsymbol{A}$ and $\boldsymbol{B}$ will be linked by a directed edge with the arrow pointing from $\boldsymbol{A}$ to $\boldsymbol{B}$. Each $V_i \in \boldsymbol{V}$ represents a set of mutually exclusive and exhaustive events, along with a joint probability distribution, P, in the knowledge base. The fundamental assumption in a Bayesian network is that the value assumed by a node is probabilistically independent of the value assumed by all other nodes in the network, except its descendants, given values of all its parents.

Fig. 1 is a very simple chain-like Bayesian network. If we are interested in the

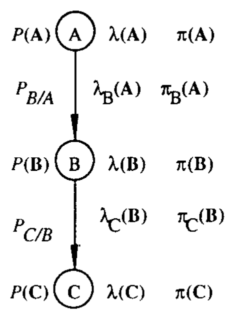

Fig. 1. A simple Bayesian network with all notation.

belief of the variables $\boldsymbol{B}$ given some observed data (evidence) for the variables $\boldsymbol{A}$ and $\boldsymbol{C}$, we will express it as

$$\begin{aligned} Bel(\boldsymbol{B}) &= P(\boldsymbol{B}/\boldsymbol{A},\boldsymbol{C}) = \alpha P(\boldsymbol{C}/\boldsymbol{A},\boldsymbol{B})P(\boldsymbol{B}/\boldsymbol{A}) \\ &= \alpha P(\boldsymbol{C}/\boldsymbol{B})P(\boldsymbol{B}/\boldsymbol{A}) = \alpha\lambda(\boldsymbol{B})\pi(\boldsymbol{B}) \end{aligned} \tag{3}$$

using Eq. (2) and noting that $P(\boldsymbol{C}/\boldsymbol{A},\boldsymbol{B}) = P(\boldsymbol{C}/\boldsymbol{B})$, since $\boldsymbol{B}$ blocks the path from $\boldsymbol{A}$ to $\boldsymbol{C}$. Eq. (3) expresses the belief of $\boldsymbol{B}$ as the product of the belief based on the evidence from the child ($\boldsymbol{C}$) and the belief based on evidence from the parent ($\boldsymbol{A}$). A detailed derivation of all other variables and link matrices can be found in Neapolitan [20] or Pearl [23].

In the computation of posterior probabilities using a Bayesian network, we assume that the variables are statistically conditionally independent given their parents. However, in practical applications, this fundamental assumption may not hold. One easy way to test this requirement is to use the Pearson's correlation coefficient for the children's distribution conditioned over the parent. For example, if there are three variables $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}\}$ and we want to test whether $\boldsymbol{B}$ and $\boldsymbol{C}$ are conditionally independent given $\boldsymbol{A}$, i.e.:

$$P(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}) = P(\boldsymbol{A})P(\boldsymbol{B}/\boldsymbol{A})P(\boldsymbol{C}/\boldsymbol{A}),$$

we use the conditional correlation coefficient ρ, which is defined for any two variables $\boldsymbol{B}$ and $\boldsymbol{C}$ given the assumed parent $\boldsymbol{A}$ as:

$$\rho = \left\| P(\boldsymbol{A})\frac{\mathrm{Cov}(\boldsymbol{B}/\boldsymbol{A},\boldsymbol{C}/\boldsymbol{A})}{\sqrt{\mathrm{Var}(\boldsymbol{B}/\boldsymbol{A})\,\mathrm{Var}(\boldsymbol{C}/\boldsymbol{A})}} \right\|, \tag{4}$$

where the $\| \ \|$ symbol means that correlation coefficient ρ is the result of summing over all the states of the variable $\boldsymbol{A}$, weighted by the probability of $\boldsymbol{A}$. When using the correlation coefficient for testing for conditional independence the ordering of the states in our variables is important and meaningful, and must reflect a monotonic progression in the measured property. If there is no meaning in the ordering of the states, we need to use a non-parametric method, such as the chi-square test, to test the conditional independence of the variables. For an example, see [15, Chapter 10.4].

A high correlation coefficient between a pair of variables, given their parent(s), indicates that the conditional independence assumption does not hold. The effect of not recognising the correlated variable can be easily illustrated by the following example.

Example. Suppose a root variable, A, has only two statistically independent sons, B and C. If the prior probability of A is $[0.2, 0.8]$ and the lambda messages from B and C are $[0.8, 0.2]$ and $[0.3, 0.7]$ respectively, then the desired posterior probabilities of A, $Bel(A)$ should be $[0.3, 0.7]$. However, if the variable of B is measured by two different processes and was included in the network as two independent variables, B and D, then we will have three lambda messages $[0.8, 0.2]$, $[0.3, 0.7]$ and $[0.8, 0.2]$ propagated to A, and the predicted posterior of $Bel(A)$ is calculated as $[0.63, 0.37]$. It is obvious that the computed posterior probability is different from the correct value.

From the above example, we see that not recognising conditionally dependent data will magnify and over-emphasise the influence of a particular piece of evidence. This will distort the posterior probabilities of the Bayesian network and may lead to misclassification.

2. Using Bayesian networks for endoscope control

We have used Bayesian networks in the practical application of building an advisory and control system for colon endoscopy. An endoscope is a medical instrument that is used for non-invasive observation of the inner surfaces of the human body. It is employed extensively in the diagnosis of colon and gastrointestinal tract diseases. When inserting an endoscope into the colon, the doctor steers the tip to the centre (which is called the lumen) and inserts it gently. If the tip is not directed correctly, and is pushed against the wall of the colon, it can be dangerous and painful for the patient. The difficulties associated with its use have meant that, to date, only highly skilled specialists are able to perform endoscopic investigations. We are proposing to rectify this situation by using computer vision to provide automatic guidance and advice during colonoscopy.

Various methods of extracting information for the navigation and advisory system have been investigated. They are represented in Fig. 2. The signal level consists of all the processing that will be performed on an image to extract all the identifiable features. The control level utilises the information from the signal level and possibly other information, such as temporal data, relationships about features, etc., in order to draw inferences. The output of this level is high-level information including advice and control information.

The signal level is made up of three algorithms. First, there is a method of identifying the lumen, or centre line of the colon, based on region extraction from

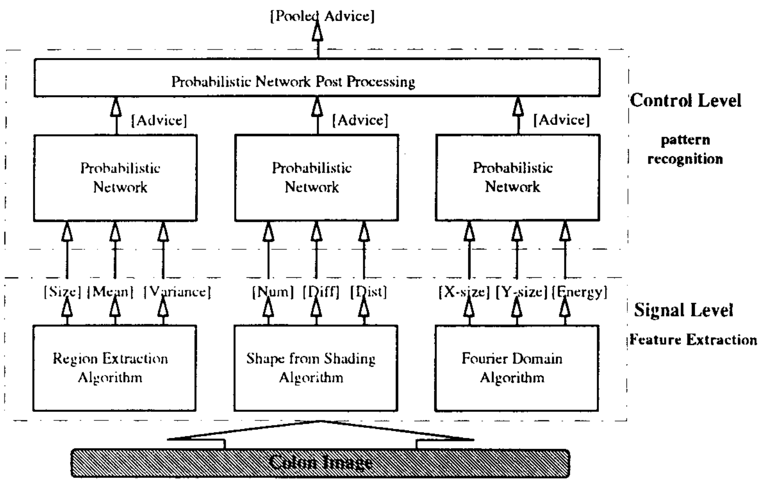

Fig. 2. Block diagram of various levels of processing in our endoscopic advisory and control system.

spatial domain data (Khan and Gillies [11,12]). Secondly, there is a shape from shading algorithm to reconstruct the surface normal $(p, q, -1)$ at a set of points in the image (Rashid and Burger [25]) and thirdly there is method which uses correlation in the Fourier domain (Kwoh and Gillies [14]).

Sucar and Gillies [32] utilised both Khan and Rashid's algorithms to implement an advisory system (control level) based on Pearl's [22] Bayesian networks. This resulted in the construction of a basic, artificial intelligence system for navigation in the colon. Although this system worked well, there were still circumstances where it failed to give correct advice. The system was improved by incorporating the information from the Fourier transform method in the Bayesian network, and the performance now seems good enough for automatic control.

Sucar observed that the advisory system mis-classified colon images in certain cases due to the fact that the conditional independence assumption of Bayesian networks had been violated. To overcome this problem, Sucar and Gillies [31] suggested the following three strategies to handle correlated data: node deletion, node combination and node creation. In practice we seldom have 100% correlated data, hence, node deletion will inevitably throw away some information. Similarly, for node combination, we have a situation where two variables must be assigned to one object. This type of situation is quite rare. The third strategy, node creation, appears to be a powerful solution method. Sucar proposed a methodological solution, namely consultation with experts, to derive a node that makes the two dependent variables conditional independent. It will in general be a very difficult process for the expert to define a function which will combine the information from the two evidence variables into a coherent variable. Hence, our current research is to devise a way to create a hidden node based on the statistical distribution of the two evidence variables and an objective function which satisfies the axiom of conditional independence. Our approach is to use training data, without seeking expert opinion, to define a mapping which will fuse the dependent information.

The sub-module for lumen identification, based on the Fourier domain, is depicted in Fig. 3. It extracts values of the following feature sets (or variables) {[X-size], [Y-size], [Energy]} from the endoscope image, for the inference of the qualitative variables [Advice] and [LDR] which indicates the presence of a large dark region (hypothesised to be the lumen) in the image. This sub-module also extracts the continuous variables {X-offset, Y-offset}, which do not appear in the network, for the estimation of the centre of the lumen to which the endoscope tip should be directed. The two propositions for the variable [Advice] are: (1) Stop, since the tip is directed at the colon wall, and (2) Advance, since there is a clear space to push the endoscope tip forward. We denote these propositions as $[\text{Advice}] \in [\text{Stop, Advance}]^{\mathrm{T}}$, where the superscript T represents the transpose. It is only possible to advance the tip if the lumen has been observed. It is perceived as a large dark region, and incorporated as an intermediate variable [LDR] under Sucar's node creation approach. We define the variable [LDR] to have two exclusive states: Not Present and Present, denoted as $[\text{LDR}] \in [\text{Not Present, Present}]^{\mathrm{T}}$. [Advice] and [LDR] are the only binary-valued variables in our system, other variables have 8 to 10 states.

Each element in the set of features {X-size, Y-size, Energy} for the inference of [LDR] and the advisory interpretation of [Advice] is a variable which has various classes or states. For example, the measured [X-size] is quantised and represented as a vector of discrete states, with its value taking any one of the eight possibilities.

$$[\text{X-size}] \in [0\text{–}8, 8\text{–}16, 16\text{–}24, 24\text{–}32, 32\text{–}40, 40\text{–}48, 48\text{–}56, 56\text{–}64]^{\mathrm{T}} .$$

We say that the variable [X-size] has the possibility of belonging to any one range of sizes defined above.

The variable [Advice] is sometimes called a proposition variable because it is a probabilistic collection of propositions. The others are simply called variables, and the individual probabilistic state for each variable is called its state or class. In

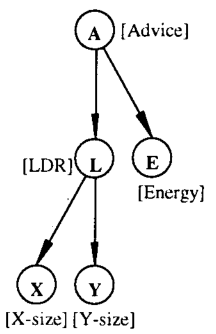

Fig. 3. The inference diagram for Fourier domain sub-module.

Bayesian networks, when a piece of evidence arrives, we will instantiate the associated variable. By instantiation, we are applying a mapping function, f of the form:

$$f([\text{X-size}]) \rightarrow [0, 1]\,,$$

$$\sum f([\text{X-size}]) = 1\,,$$

where f is actually a probability of the variable being in a specific state. (Note that in cases where we have inconclusive evidence that a variable takes a certain value, we may use virtual evidence (Neapolitan [20]) which does not sum up to one. It can be normalised and combined with other evidence to obtain the posterior probability.) In our application, we always normalise our evidence to satisfy the axioms of probability.

An alternative approach to the above problem is to use neural network methods. However, we have a number of reasons to prefer Bayesian networks for our application. First, the output is the posterior probability of the hypothesis, given the input, which provides a natural mechanism to draw inferences and select the best advice. Secondly, Bayesian networks can be trained branch by branch. When additional information is obtained for the model, it can be incorporated into the existing Bayesian network without retraining, provided it does not break the fundamental assumption of conditional independence. Thirdly, Bayesian networks can incorporate both subjective probabilities [24] and objective probabilities. In the subjective approach, probability is regarded as the degree of belief of a particular individual. In the objective approach, probability has nothing to do with human beliefs, and manifests itself in the observed statistical frequency with which a particular event occurs. Neural networks only work with objective probabilities, whereas Bayesian networks can combine subjective and objective information. Fourthly, Bayesian networks cope well with partial information. They have a very strong normative theory to manage ignorance when compared to other connectionist networks commonly used for classification. Lastly, the multi-layer neural network is a fully connected network in which a collection of linear regression models store statistical information within the connecting weights. A neural network is a chain-like Bayesian network with the addition of an activation function and a nonlinear squashing function. Hence, for any problem it will have many more parameters than the equivalent Bayesian network. For example the hand-written character recognition system of Le Cun et al. [18] has 2600 parameters. In comparison, Chow and Liu [1], can model the same domain using a semantic structure similar to a Bayesian network with considerably fewer parameters. Although there are many ways to reduce the number of parameters (Le Cun et al. [18], Weigend et al. [33]), we believe that it is better to use the model which requires fewer parameters.

Denker and Le Cun [3] recognised the importance of the classification problem, in neural networks, which have mutually exclusive outcomes that satisfy the axioms of probability. They satisfied this constraint by treating a trained neural network as a pre-processor that produces a feature vector which is further processed by classical statistical estimation techniques. They assumed the output vector of the neural network, $\underline{O}_p$ (which is represented as a variable as $\boldsymbol{O}_p$), given the input vector set of all the training data, $\{\underline{I}_1, \underline{I}_2, \underline{I}_3 \ldots\}$ (represented as a variable $\boldsymbol{I}$), can be approximated by a normal distribution. They derived expressions to calculate the first two moments of the probability distribution $P(\boldsymbol{O}_p/\boldsymbol{I})$, and used them to adjust the weights of the neural networks to achieve a Bayesian-like output, where the numerical value approximates the posterior probability of the output variable given the observed evidence (inputs). This type of system will not give a true Bayesian output and will perform badly in regions where the inputs have low probability for all states of the desired variable (root node) as reported by Richard and Lippmann [26]. This may be due to the nonlinear sigmoid mapping function, used in neural networks, which forces the output to be effectively bipolar.

In summary, we believe that, where possible, it is much more efficient and economical to build structures rather than use a general connectionist network. If there are constraints that cannot be fully satisfied statistically, we then seek some strategy to modify the structure to improve the quality of the model.

Once we have determined the structure, or inference diagram, of the Bayesian network, we can obtain an initial estimate of the conditional probabilities of the link matrices for each edge in the Bayesian network by the QUALQUANT methodology [31]. We then test the assumption of conditional independence. If the criterion is not satisfied, we create a hidden node to form a star structure and express the error cost function ξ with respect to the optimal values. If we expand the expression for those conditional probabilities by Taylor's series, we can use a gradient search method, such as the least mean squares algorithm (Widrow and Stearns [34]) to find solutions by an iterative process:

$$\begin{aligned}\underline{\underline{P(n+1)}} &= \underline{\underline{P(n)}} - \eta\, \Delta\underline{\underline{P(n)}} \\ &= \underline{\underline{P(n)}} - \eta E\left[\frac{\partial \xi}{\partial \underline{\underline{P(n)}}}\right]\end{aligned} \tag{5}$$

or in a more complex error correction form such as:

$$\underline{\underline{P(n+1)}} = \underline{\underline{P(n)}} - \eta(1-\alpha)\, \Delta\underline{\underline{P(n)}} + \alpha\, \Delta\underline{\underline{P(n-1)}}\,. \tag{6}$$

In the above equations, η is called the learning rate or the relaxation constant, which is typically a small value in the range of 0.01 to 0.1. Widrow and Stearns [34] have worked out an upper bound on the learning rate for linear models. The second term in Eq. (6) is called the momentum term and α takes a value within the range $[0, 1]$. This term is commonly added in neural network training algorithms. Notice that we are adjusting the parameter values in the negative gradient direction, because we are searching for the minimum of the error cost function.

3. Introducing hidden nodes into Bayesian networks

Pearl [22] proposed the use of a hidden variable to satisfy the axioms of conditional independence. By adding a hidden variable between correlated variables and their parent, he created a star-decomposable structure. In his work, he dealt with star decomposition for binary-valued random variables, say $\{V_1, V_2, V_3, \ldots\}$ with a given probability mass function $P(V)$, and obtained a closed form solution for a variable W that will satisfy the equation

$$P(V, W) = \prod_{V_i \subset V} P(V_i / W) P(W) .$$

Xu and Pearl [33] extended the ideas in Pearl [22] to utilise multi-variate Gaussian variables. However, in our application, most of our variables are not binary valued nor Gaussian, and so we need to derive an algorithm that can handle multi-valued variables some of which are unobservable (hidden nodes). A similar problem of configuring probabilistic models for bi-valued hypotheses with hidden variables was mentioned by Hinton et al. [10] as a task that a Boltzmann machine should be able to solve. Further details on how to do this have been proposed by Neal [19] who implemented the Boltzmann machine with multiple restarts, bindary variables and signal propagation in only one direction. Another precursor of our work is the EM algorithm [3] which consists of the expectation step (E-step to estimate the complete data from the observed (incomplete) data) and the maximisation step (M-step to maximise the likelihood). Lauritzen [16] and Spiegelhalter et al. [29] have experimented the EM algorithm to update the parameters in the Bayesian networks. They concluded that the likelihood function has a number of local maxima and straight maximum likelihood gives results with unsuitably extreme probabilities. Furthermore, the EM algorithm is known to converge relatively slowly when it is getting close to a maximum. They suggested that algorithms exploiting the gradient could be preferable. Paass [21] describes a stochastic version of the EM algorithm which can be considered a modified version of the Boltzmann machine.

For the network of Fig. 1, if we have the desired posterior probability of the Bayesian network, given some instantiated evidence, we can define a monotonic error cost function for the differences between the desired posterior probability and the estimated posterior probability. We can use this to devise a strategy to adjust the conditional probability matrices in the links belonging to a hidden node. If the conditional independence criteria are satisfied, we should expect a minimum of the error cost function throughout the training set.

Let the variable $\boldsymbol{A}$ have A states and a set of training data be denoted $\boldsymbol{E} \in \Im$. If we are interested in the posterior probability (belief) of $\boldsymbol{A}$ given some evidence $\boldsymbol{E}$, denoted as $Bel(\boldsymbol{A})$, then we express $Bel(\boldsymbol{A})$ as a nonlinear function of all the evidence

$$Bel(\boldsymbol{A}) = f(\boldsymbol{E}) .$$

The expectation of the squared-error cost function, Δ, is

$$\Delta = E\{\xi\} = E\left\{\sum_{i=1}^{A} [D(a_i) - Bel(a_i)]^2\right\} \tag{7}$$

where $E\{\cdot\}$ is the expectation operator, and $D(a_i)$ is the desired value of a_i. Using the joint probability of the input and the desired output, Hampshire and Pearlmutter [8] show that when the network parameters are chosen to minimise a squared-error cost function, the outputs are the conditional expectations of the desired outputs which minimise the mean-squared estimation error. In their paper, they refer to neural networks, but their derivation can be easily generalised to any type of network in which the input and output relationship can be expressed by a mapping function.

$$\min\{\Delta\} = \min\left\{E\left\{\sum_{i=1}^{A} [E\{D(a_i)/\boldsymbol{E}\} - (Bel(z_i))]^2\right\}\right\}. \tag{8}$$

The justification of minimising the mean-squared error cost function is presented in Appendix A. In Eq. (8), $E\{D(a_i)/\boldsymbol{E}\}$ is the expected belief of $\boldsymbol{A}$ given the evidence $\boldsymbol{E}$, and

$$E\{D(a_i)/\boldsymbol{E}\} = \sum_{j=1}^{A} D(a_i)P(a_j/\boldsymbol{E}) \tag{9}$$

is the conditional probability of desired states given the evidence $\boldsymbol{E}$, weighted by the instantiated value of the states vector. If only one of the states, i, is assigned the value one and the rest zeros, Eq. (9) will become:

$$E\{D(a_i)/\boldsymbol{E}\} = P(a_i/\boldsymbol{E})$$

which is the Bayesian probability. If more than one state is instantiated with value greater than zero, Eq. (9) will become a dot product of the instantiated value and the Bayesian probability vector, which satisfies the axioms of virtual evidence. This provides us with additional insights to the least mean squares algorithm. Contrary to popular belief that the training data of the desired variable must be one of N cases, we can provide better training using virtual evidence if we are not sure in exactly which state the desired output should be instantiated. The only caution that we must observe is that the sum of all instantiated values must be equal to unity. For example in our application, for a particular colon image, the expert's advice can be a vector of $[\text{Stop} = 0, \text{Advance} = 1]^{\mathrm{T}}$ or a less definite advice of $[\text{Stop} = 0.2, \text{Advance} = 0.8]^{\mathrm{T}}$.

Note that ξ is a scalar and it is always positive. ξ is known as the squared-error cost function which is used to measure the difference between the actual outputs and the desired output. It is the most widely used cost function of all the possibilities. Others can have slightly different effects, for example, the cross-entropy cost function (which is also known as Kullback–Lieber (KL) distortion [13, 28]) weights errors more heavily when the actual outputs are near zero and one.

$$\xi = \sum_{i=1}^{A} [D(a_i)\log(D(a_i)/Bel(a_i))] .$$

Hampshire and Pearlmutter [8] have shown that the objective functions to minimise the squared-error cost function and cross-entropy cost functions are asymptotically equal. That means that if we have a large training data set that is a good representation of the problem, it really does not matter whether we express our error function as a squared-error cost function, Eq. (7), or as a cross-entropy cost function.

To obtain the derivation of the gradients for the conditional probabilities of the hidden node, we start with the most simple star structure shown in Fig. 4.

We first write the operating equations [20] from the leaf nodes to the root node. Given that:

- $\pi(a_i)$ is the prior probability distribution of the variable, $P(A)$, which can be obtained from the domain of training data,
- $D(a_i)$, is the desired value for the root variable (A) for each set of training data,
- $\lambda(c_k)$ is the evidence of the instantiated leaf node C, and
- $\lambda(d_l)$ is the evidence of the instantiated node D,

we have:

$$\lambda_C(h_j) = \sum_{k=1}^{C} P(c_k/h_j)\lambda(c_k) , \tag{10}$$

$$\lambda_D(h_j) = \sum_{l=1}^{D} P(d_l/h_j)\lambda(d_l) , \tag{11}$$

$$\lambda(h_j) = \lambda_C(h_j)\lambda_D(h_j) , \tag{12}$$

$$\lambda_H(a_i) = \sum_{j=1}^{H} P(h_j/a_i)\lambda(h_j) , \tag{13}$$

$$\begin{aligned} &\lambda(a_i) = \lambda_H(a_i) , \\ &bel(a_i) = \lambda(a_i)\pi(a_i) , \end{aligned} \tag{14}$$

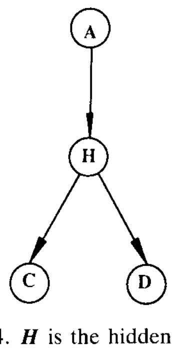

Fig. 4. H is the hidden node.

$$Bel(a_i) = \frac{bel(a_i)}{\sum_{m=1}^{A} bel(a_m)}, \tag{15}$$

$$e(a_i) = D(a_i) - Bel(a_i), \tag{16}$$

$$\xi = \sum_{i=1}^{A} e(a_i)^2. \tag{17}$$

The objective function to be minimised is the expectation of (17). To do that we must obtain the gradients of (17) with respect to all the conditional probabilities in the link matrix. Choosing a general element $P(h_j/a_i)$ from the link matrix between node $\boldsymbol{A}$ and $\boldsymbol{H}$, we expand the gradient using the chain rule for partial derivatives to get,

$$\frac{\partial \xi}{\partial P(h_j/a_i)} = \frac{\partial \xi}{\partial e(a_m)} \frac{\partial e(a_m)}{\partial Bel(a_m)} \frac{\partial Bel(a_m)}{\partial bel(a_i)} \frac{\partial bel(a_i)}{\partial \lambda(a_i)} \frac{\partial \lambda(a_i)}{\partial P(h_j/a_i)} \tag{18}$$

where the individual terms in the expansion are

$$\frac{\partial \xi}{\partial e(a_m)} = 2e(a_m),$$

$$\frac{\partial e(a_m)}{\partial Bel(a_m)} = -1,$$

$$\begin{aligned}\frac{\partial Bel(a_m)}{\partial bel(a_i)} &= \frac{\delta_{im}}{\sum_{m=1}^{A} bel(a_m)} - \frac{bel(a_i)}{(\sum_{m=1}^{A} bel(a_m))^2} \\ &= \frac{1}{\sum_{m=1}^{A} bel(a_m)}(\delta_{im} - Bel(a_i)),\end{aligned}$$

where

$$\delta_{im} = \begin{cases} 1, & i = m, \\ 0, & i \neq m, \end{cases}$$

$$\frac{\partial bel(a_i)}{\partial \lambda(a_i)} = \pi(a_i),$$

$$\frac{\partial \lambda(a_i)}{\partial P(h_j/a_i)} = \lambda(h_j).$$

Notice that all the elements of $\partial Bel(a_m)/\partial bel(a_i)$ form a matrix, hence when we put all the partial derivatives together, we must sum over all the values indexed by the subscript m. This is because the value of Bel is dependent on all other values of bel before normalisation.

Substituting into Eq. (18) we get

$$\frac{\partial \xi}{\partial P(h_j/a_i)} = \sum_{m=1}^{A} \left[-2e(a_m)\left(\frac{\delta_{im}}{\sum_{m=1}^{A} bel(a_m)} - \frac{bel(a_i)}{(\sum_{m=1}^{A} bel(a_m))^2} \right) \right] \pi(a_i)\lambda(h_j)$$
$$= \frac{-2}{\sum_{m=1}^{A} bel(a_m)} \sum_{m=1}^{A} [e(a_m)(\delta_{im} - Bel(a_i))]\pi(a_i)\lambda(h_j)\,. \tag{19}$$

If we do the partial derivative for each conditional probability in the links between node $\boldsymbol{A}$ and node $\boldsymbol{H}$ and form them into a matrix, we will have the gradient matrix which is required to adjust the conditional probabilities of the link.

Similarly for an element in $(\partial\xi)/(\partial P(c_k/h_j))$, we have

$$\frac{\partial \xi}{\partial P(c_k/h_j)} = \frac{\partial \xi}{\partial \lambda(h_j)} \frac{\partial \lambda(h_j)}{\partial \lambda_C(h_j)} \frac{\partial \lambda_C(h_j)}{\partial P(c_k/h_j)}\,. \tag{20}$$

When we expand the $(\partial\xi)/(\partial\lambda(h_j))$, we must do so under a summation for all states of a_i, because the change in the conditional probabilities will effect the result in every state of belief in $\boldsymbol{A}$.

$$\frac{\partial \xi}{\partial P(c_k/h_j)} = \left[\sum_{i=1}^{A} \frac{\partial \xi}{\partial e(a_m)} \frac{\partial e(a_m)}{\partial Bel(a_m)} \frac{\partial Bel(a_m)}{\partial bel(a_i)} \frac{\partial bel(a_i)}{\partial \lambda(a_i)} \frac{\partial \lambda(a_i)}{\partial \lambda_H(a_i)} \frac{\partial \lambda_H(a_i)}{\partial \lambda(h_j)} \right]$$
$$\times \frac{\partial \lambda(h_j)}{\partial \lambda_C(h_j)} \frac{\partial \lambda_C(h_j)}{\partial P(c_k/h_j)}$$
$$= \left\{ \sum_{i=1}^{A} \left(\frac{-2}{\sum_{m=1}^{A} bel(a_m)} \sum_{m=1}^{A} [e(a_m)(\delta_{im} - Bel(a_i))] \right) P(h_j/a_i) \right\}$$
$$\times \frac{\partial \lambda(h_j)}{\partial \lambda_C(h_j)} \frac{\partial \lambda_C(h_j)}{\partial P(c_k/h_j)}$$
$$= \frac{-2}{\sum_{m=1}^{A} bel(a_m)} \left\{ \sum_{i=1}^{A} \left(\sum_{m=1}^{A} [e(a_m)(\delta_{im} - Bel(a_i))] \right) P(h_j/a_i) \right\}$$
$$\times \lambda_D(h_j)\lambda(c_k)\,. \tag{21}$$

Similarly,

$$\frac{\partial \xi}{\partial P(d_l/h_j)} = \left[\sum_{i=1}^{A} \frac{\partial \xi}{\partial e(a_m)} \frac{\partial e(a_m)}{\partial Bel(a_m)} \frac{\partial Bel(a_m)}{\partial bel(a_i)} \frac{\partial bel(a_i)}{\partial \lambda(a_i)} \frac{\partial \lambda(a_i)}{\partial \lambda_H(a_i)} \frac{\partial \lambda_H(a_i)}{\partial \lambda(h_j)} \right]$$
$$\times \frac{\partial \lambda(h_j)}{\partial \lambda_D(h_j)} \frac{\partial \lambda_D(h_j)}{\partial P(d_l/h_j)}$$
$$= \frac{-2}{\sum_{m=1}^{A} bel(a_m)} \left\{ \sum_{i=1}^{A} \left(\sum_{m=1}^{A} [e(a_m)(\delta_{im} - Bel(a_i))] \right) P(h_j/a_i) \right\}$$
$$\times \lambda_C(h_j)\lambda(d_l)\,. \tag{22}$$

Note that, from Eqs. (19), (21) and (22), it is not difficult to show that the elements of the Hessian (the second derivatives of all the conditional probabilities) are semi-positive definite. Hence, we will move towards the minimum of the performance surface of the error function defined over the conditional probabilities.

4. Matrix representation of the gradient for the squared-error cost function for backward propagation

The derivation in Section 3 for individual elements of the link matrices results in a rather cumbersome formulation in which it is difficult to visualise the process. Furthermore, the multiple summations in the expressions make programming and representation a rather tedious task. A better formulation can be obtained by exploiting the elegant representation and implicit summation of matrix multiplication in the derivation of the gradient matrices.

Notation

In all our equation derivations, we are using column vectors, such as

$$\pi(\underline{A}) = \begin{bmatrix} \pi(a_1) \\ \pi(a_2) \\ \vdots \\ \pi(a_A) \end{bmatrix}.$$

For ease of notation, we will use $\underline{A}$ to indicate a column vector with states $[a_1, a_2, \ldots, a_A]^{\mathrm{T}}$, and A without the underscore, to indicate the dimension of $\underline{A}$. The superscript T, as usual, denotes the transpose.

For a matrix, we use the notation

$$P_{\underline{H/A}} = \begin{bmatrix} P(h_1/a_1) & P(h_2/a_1) & \cdots & P(h_H/a_1) \\ P(h_1/a_2) & P(h_2/a_2) & \cdots & P(h_H/a_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(h_1/a_A) & P(h_2/a_A) & \cdots & P(h_H/a_A) \end{bmatrix}$$

where A is dimension($\underline{A}$) and H is dimension($\underline{H}$). The two variable underscore indicates a matrix of dimension A times H.

We introduce the operator $\otimes$ to indicate element-by-element multiplication. The two operand matrices (or vectors) must be of the same size. The result is a matrix (or vector) of the original dimension, where each element is the product of the elements in the corresponding positions in the operands. This operator is useful when expressing the fusion of lambda messages and the fusion of pi and lambda values.

We use the following rules, which the reader may verify for himself, for matrix/vector differentiation:

(1) If we differentiate a scalar by a vector, the resultant is a vector of the same dimension as the vector.

(2) If we differentiate a vector by a vector of same dimension we would normally obtain a matrix. However, in the case where two vectors are combined by the term-by-term multiplication operator $\otimes$, and the differentiation is taken with respect to one of the vectors, the result is a vector.

(3) If a vector $\underline{A}$, which is obtained by post-multiplying a matrix $\boldsymbol{M}$ (dimension A times H) with a vector $\underline{H}$, is differentiated with respect to the second vector $\underline{H}$, we will obtain the matrix $\boldsymbol{M}$. If $\underline{A}$ is combined with a vector $\underline{C}$ (dimension A) using the operator $\otimes$ in the original equation, then we will get the same result by pre-multiplying the matrix $\boldsymbol{M}$ with the transpose of vector $\underline{C}$.

(4) If a vector $\underline{A}$, which is obtained by post-multiplying a matrix $\boldsymbol{M}$ (dimension A times H) with a vector $\underline{H}$, is differentiated by the matrix $\boldsymbol{M}$, the result is a matrix $\boldsymbol{N}$ where each row is the transpose of $\underline{H}$. If $\underline{A}$ is to be combined with a vector $\underline{C}$ (dimension A) using the operator $\otimes$ in the original equation, then we can obtain the result taking the outer product of vector $\underline{C}$ and the transpose of vector $\underline{H}$ to produce a matrix of dimension A times H.

Before we express all the operating equations for the Bayesian network using matrices and vectors, we require, as previously, the following data in vector notation:

- $\pi(\underline{A})$ is the prior probability distribution of $\underline{A}$, $P(\boldsymbol{A})$, which can be obtained from the domain of training data,
- $D(\underline{A})$ is the desired state for the root node ($\boldsymbol{A}$) for each set of training data,
- $\lambda(\underline{C})$ is the evidence of the instantiated node $\boldsymbol{C}$, and
- $\lambda(\underline{D})$ is the evidence of the instantiated node $\boldsymbol{D}$.

The operating equations are,

$$\lambda_C(\underline{H}) = P_{\underline{C/H}}\lambda(\underline{C})\,, \tag{23}$$

$$\lambda_D(\underline{H}) \cdot P_{\underline{D/H}}\lambda(\underline{D})\,, \tag{24}$$

$$\begin{aligned} \lambda(\underline{H}) &= \lambda_C(\underline{H}) \otimes \lambda_D(\underline{H})\,, \\ bel(\underline{H}) &= \lambda(\underline{H}) \otimes \pi(\underline{H})\,, \end{aligned} \tag{25}$$

$$\lambda_H(\underline{A}) = P_{\underline{H/A}}[bel(\underline{H}) \backslash \pi(\underline{H})] = P_{\underline{H/A}}\lambda(\underline{H})\,. \tag{26}$$

In Eq. (26), the lambda message to the parent is the belief value excluding the contribution from the parent itself. It is the same as the lambda value in the case where there is only a single parent.

$$\begin{aligned} \lambda(\underline{A}) &= \lambda_H(\underline{A})\,, \\ bel(\underline{A}) &= \lambda(\underline{A}) \otimes \pi(\underline{A})\,, \end{aligned} \tag{27}$$

$$Bel(\underline{A}) = \frac{bel(\underline{A})}{|bel(\underline{A})|}\,, \tag{28}$$

$$e(\underline{A}) = D(\underline{A}) - Bel(\underline{A})\,. \tag{29}$$

where $bel()$ is the un-normalised form of the belief function and $|bel(\underline{A})|$ is the norm of $bel(\underline{A})$. $D(\underline{A})$ is the desired result of node A.

$$\xi = e(\underline{A})^{\mathrm{T}} e(\underline{A})\,. \tag{30}$$

Note that ξ is a scalar and it is always positive.

Now, to do the partial differentiation, we again make use of the chain rule

$$\begin{aligned}\frac{\partial \xi}{\partial P_{\underline{H/A}}} &= \frac{\partial \xi}{\partial e(\underline{A})}\frac{\partial e(\underline{A})}{\partial Bel(\underline{A})}\frac{\partial Bel(\underline{A})}{\partial bel(\underline{A})}\frac{\partial bel(\underline{A})}{\partial \lambda(\underline{A})}\frac{\partial \lambda(\underline{A})}{\partial \lambda_H(\underline{A})}\frac{\partial \lambda_H(\underline{A})}{\partial P_{\underline{H/A}}} \\ &= \left[\left[\frac{\partial \xi}{\partial e(\underline{A})}\frac{\partial e(\underline{A})}{\partial Bel(\underline{A})}\frac{\partial Bel(\underline{A})}{\partial bel(\underline{A})}\right]\frac{\partial bel(\underline{A})}{\partial \lambda(\underline{A})}\frac{\partial \lambda(\underline{A})}{\partial \lambda_H(\underline{A})}\right]\frac{\partial \lambda_H(\underline{A})}{\partial P_{\underline{H/A}}}\,.\end{aligned} \tag{31}$$

Care should be taken with the matrix operators used between the matrices when substituting for the partial derivatives in Eq. (31). To clarify the meaning we use $\boldsymbol{I}$ to mean an identity matrix of dimension A, and the notation $ones(\underline{A})^{\mathrm{T}}$ to be a row vector of dimension A, with each entry equal to 1. Making the substitutions equivalent to those defined in Section 3 we obtain:

$$\begin{aligned}\frac{\partial \xi}{\partial P_{\underline{H/A}}} &= \left\{\left[(-2e(\underline{A}))^{\mathrm{T}}\left(\frac{1}{|bel(\underline{A})|}[I - Bel(\underline{A})ones(\underline{A})^{\mathrm{T}}]\right)\right]^{\mathrm{T}} \otimes \pi(\underline{A})\right\}\lambda(\underline{H})^{\mathrm{T}} \\ &= \frac{-2}{|bel(\underline{A})|}\{[e(\underline{A})^{\mathrm{T}}(I - Bel(\underline{A})ones(\underline{A})^{\mathrm{T}})]^{\mathrm{T}} \otimes \pi(\underline{A})\}\lambda(\underline{H})^{\mathrm{T}}\,.\end{aligned} \tag{32}$$

Notice that the summation of (32) is implemented by matrix multiplication of $[e(\underline{A})^{\mathrm{T}}(I - Bel(\underline{A})ones(A)^{\mathrm{T}})]$ to produce a row vector of size A times 1. From Eq. (32), it is easy to see that $(\partial\xi)/(\partial\lambda_H(\underline{A}))$ behaves like the pi message from $\boldsymbol{A}$ to $\boldsymbol{H}$ and has dimension A.

For the link matrix between node $\boldsymbol{H}$ and $\boldsymbol{C}$, we have

$$\begin{aligned}\frac{\partial \xi}{\partial P_{\underline{C/H}}} &= \frac{\partial \xi}{\partial e(\underline{A})}\frac{\partial e(\underline{A})}{\partial Bel(\underline{A})}\frac{\partial Bel(\underline{A})}{\partial bel(\underline{A})}\frac{\partial bel(\underline{A})}{\partial \lambda(\underline{A})}\frac{\partial \lambda(\underline{A})}{\partial \lambda_H(\underline{A})}\frac{\partial \lambda_H(\underline{A})}{\partial \lambda(\underline{H})} \\ &\quad\times \frac{\partial \lambda(\underline{H})}{\partial \lambda_C(\underline{H})}\frac{\partial \lambda_C(\underline{H})}{\partial P_{\underline{C/H}}} \\ &= \left\{\left[\frac{\partial \xi}{\partial \lambda_H(\underline{A})}\right]\frac{\partial \lambda_H(\underline{A})}{\partial \lambda(\underline{H})}\frac{\partial \lambda(\underline{H})}{\partial \lambda_C(\underline{H})}\right\}\frac{\partial \lambda_C(\underline{H})}{\partial P_{\underline{C/H}}}\,.\end{aligned} \tag{33}$$

From Eq. (33), we have

$$\frac{\partial \lambda_H(\underline{A})}{\partial \lambda(\underline{H})} = P_{\underline{H/A}} \quad \text{and} \quad \frac{\partial \lambda(\underline{H})}{\partial \lambda_C(\underline{H})} = \lambda_D(\underline{H})$$

which is analogous to the way that the pi message from $\boldsymbol{H}$ to $\boldsymbol{C}$ will be derived, by "resizing" the evidential information from the parent node and including all the evidential information from the other siblings. Thus, $(\partial\xi)/(\partial\lambda_C(\underline{H}))$ should be thought of as a gradient message from $\boldsymbol{H}$ to $\boldsymbol{C}$.

$$\frac{\partial\xi}{\partial\lambda_C(\underline{H})} = \left[P^{\mathrm{T}}_{\underline{H/A}} \frac{\partial\xi}{\partial\lambda_H(\underline{A})} \right] \otimes \lambda_D(\underline{H}) \tag{34}$$

and the corresponding pi message from $\boldsymbol{H}$ to $\boldsymbol{C}$ if $\boldsymbol{A}$ and $\boldsymbol{D}$ are instantiated will be

$$\pi_C(\underline{H}) = [P^{\mathrm{T}}_{\underline{H/A}} \pi_H(\underline{A})] \otimes \lambda_D(\underline{H}) \,. \tag{35}$$

The reader will now probably recognise that gradient updating through gradient propagation is analogous to belief updating when using the network; each gradient message replacing the corresponding pi message or lambda message in the link. Hence,

$$\frac{\partial\xi}{\partial P_{\underline{C/H}}} = \frac{-2}{|bel(\underline{A})|} [\{P^{\mathrm{T}}_{\underline{H/A}}[\{[e(\underline{A})^{\mathrm{T}}(I - Bel(\underline{A})ones(\underline{A})^{\mathrm{T}})]^{\mathrm{T}} \otimes \pi(\underline{A})\}]\} \\ \otimes \lambda_D(\underline{H})]\lambda(\underline{C})^{\mathrm{T}} \tag{36}$$

and similarly,

$$\frac{\partial\xi}{\partial P_{\underline{D/H}}} = \frac{-2}{|bel(\underline{A})|} [\{P^{\mathrm{T}}_{\underline{H/A}}[\{[e(\underline{A})^{\mathrm{T}}(I - Bel(\underline{A})ones(\underline{A})^{\mathrm{T}})]^{\mathrm{T}} \otimes \pi(\underline{A})\}]\} \\ \otimes \lambda_C(\underline{H})]\lambda(\underline{D})^{\mathrm{T}} \tag{37}$$

Comparing the result that we have just obtained with the equations of Section 3, it is easy to see that they are the same. The matrix formulation is very clear, elegant and easy to program. The expected gradient for each epoch is

$$E\left[\frac{\partial\xi}{\partial P}\right] = \frac{1}{p}\sum_{k=1}^{p} \frac{\partial\xi}{\partial P} \,. \tag{38}$$

5. Derivation of the gradient for forward propagation

In Section 4 we derived the equations for the case where the change in the conditional probabilities is propagated from the evidence nodes to the root node. In Bayesian networks, there is no strict direction of signal flow, and so queries can be made at any node in the network. Thus the training data set $\boldsymbol{X}$ is an unordered collective $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \ldots, \boldsymbol{X}_{|X|}\}$. Since Bayesian networks must be able to handle partial evidence, where some nodes remain un-instantiated, we can define the cost function to encompass all situations as follows:

$$\min_{\Delta}\left\{ \Delta = \sum_{i=1}^{|X|} \sum_{Z \subseteq X \setminus X_i} E\left\{ \sum_{j=1}^{|X_i|} [D(\boldsymbol{X}_{ij}) - Bel(\boldsymbol{X}_{ij}/\boldsymbol{Z})]^2 \right\} \right\}$$

In our experiments we found that, if we define our cost error function to encompass both back propagation, where we instantiate the leaf nodes, and forward propagation, where we instantiate the root node, we can achieve very good results. We will now give the derivation for propagating the delta change to the conditional probabilities in the forward direction.

For forward propagation, we have the following known data:

- $\pi(\underline{A})$ which is the instantiated value for root node $\boldsymbol{A}$, designated $D(\underline{A})$ in Section 3;
- $D(\underline{C})$ is the expected partial belief of node $\boldsymbol{C}$, which was $\lambda(\underline{C})$ in Section 4;
- $D(\underline{D})$ is the expected partial belief of node $\boldsymbol{D}$, which was $\lambda(\underline{D})$ in Section 4, and
- $\lambda(\underline{C})$ and $\lambda(\underline{D})$ are unit vectors of dimensions C and D respectively.

The partial derivatives for forward propagation can be found in a similar manner to the backward propagation derivatives given in Section 4.

$$\begin{aligned} &\pi(\underline{A}) = D(\underline{A}),\\ &\pi_H(\underline{A}) = \pi(\underline{A}),\\ &\lambda_C(\underline{H}) = P_{\underline{C/H}}\lambda(\underline{C}),\\ &\lambda_D(\underline{H}) = P_{\underline{D/H}}\lambda(\underline{D}). \end{aligned} \tag{39}$$

However, since both $\lambda(\underline{C})$ and $\lambda(\underline{D})$ are not instantiated, $\lambda_C(\underline{H})$ and $\lambda_D(\underline{H})$ can be shown to be unit vectors of dimension H.

$$\begin{aligned} &\pi(\underline{H}) = P^{\mathrm{T}}_{\underline{H/A}}\pi_H(\underline{A}),\\ &\lambda(\underline{H}) = \lambda_C(\underline{H}) \otimes \lambda_D(\underline{H}),\\ &Bel(\underline{H}) = \frac{\pi(\underline{H}) \otimes \lambda(\underline{H})}{\pi(\underline{H})\lambda(\underline{H})^{\mathrm{T}}} = \pi(\underline{H}), \end{aligned} \tag{40}$$

$$\begin{aligned} &\pi_C(\underline{H}) = \frac{Bel(\underline{H})}{\lambda_C(\underline{H})} = \pi(\underline{H}),\\ &\pi_D(\underline{H}) = \frac{Bel(\underline{H})}{\lambda_D(\underline{H})} = \pi(\underline{H}), \end{aligned} \tag{41}$$

$$\pi(\underline{C}) = P^{\mathrm{T}}_{\underline{C/H}}\pi_C(\underline{H}), \tag{42}$$

$$\pi(\underline{D}) = P^{\mathrm{T}}_{\underline{D/H}}\pi_D(\underline{H}), \tag{43}$$

$$\begin{aligned} &Bel(\underline{C}) = \pi(\underline{C}),\\ &Bel(\underline{D}) = \pi(\underline{D}). \end{aligned} \tag{44}$$

Since we are not instantiating all the leaf nodes their λ values are all unity. Thus we have the expressions

$$\begin{aligned} &e(\underline{C}) = D(\underline{C}) - Bel(\underline{C}) = D(\underline{C}) - \pi(\underline{C}),\\ &e(\underline{D}) = D(\underline{D}) - Bel(\underline{D}) = D(\underline{D}) - \pi(\underline{D}). \end{aligned} \tag{45}$$

We define the error function (which is a scalar) as the sum of the total errors for all states in all leaf nodes.

$$\xi = e(\underline{C})^{\mathrm{T}} e(\underline{C}) + e(\underline{D})^{\mathrm{T}} e(\underline{D}) \tag{46}$$

and perform the partial differentiation as before

$$\begin{aligned}
\frac{\partial \xi}{\partial P_{\underline{C/H}}} &= \frac{\partial \xi}{\partial e(\underline{C})} \frac{\partial e(\underline{C})}{\partial Bel(\underline{C})} \frac{\partial Bel(\underline{C})}{\partial \pi(\underline{C})} \frac{\partial \pi(\underline{C})}{\partial P_{\underline{C/H}}} \\
&= \left[\frac{\partial \xi}{\partial e(\underline{C})} \frac{\partial e(\underline{C})}{\partial Bel(\underline{C})} \frac{\partial Bel(\underline{C})}{\partial \pi(\underline{C})} \right] \frac{\partial \pi(\underline{C})}{\partial P_{\underline{C/H}}} \\
&= -2\pi_C(\underline{H}) e(\underline{C})^{\mathrm{T}} \\
&= -2[P^{\mathrm{T}}_{\underline{H/A}} \pi(\underline{A})] e(\underline{C})^{\mathrm{T}},
\end{aligned} \tag{47}$$

$$\begin{aligned}
\frac{\partial \xi}{\partial P_{\underline{D/H}}} &= \frac{\partial \xi}{\partial e(\underline{D})} \frac{\partial e(\underline{D})}{\partial Bel(\underline{D})} \frac{\partial Bel(\underline{D})}{\partial \pi(\underline{D})} \frac{\partial \pi(\underline{D})}{\partial P_{\underline{D/H}}} \\
&= -2[P^{\mathrm{T}}_{\underline{H/A}} \pi(\underline{A})] e(\underline{D})^{\mathrm{T}},
\end{aligned} \tag{48}$$

$$\begin{aligned}
\frac{\partial \xi}{\partial P_{\underline{H/A}}} &= \sum_{x \in \{C,D\}} \left\{ \left[\left[\frac{\partial \xi}{\partial \pi(\underline{x})} \right] \frac{\partial \pi(\underline{x})}{\partial \pi_x(\underline{H})} \right] \frac{\partial \pi_x(\underline{H})}{\partial \pi(\underline{H})} \right\} \frac{\partial \pi(\underline{H})}{\partial P_{\underline{H/A}}} \\
&= \frac{\partial \xi}{\partial e(\underline{C})} \frac{\partial e(\underline{C})}{\partial \pi(\underline{C})} \frac{\partial \pi(\underline{C})}{\partial \pi_C(\underline{H})} \frac{\partial \pi_C(\underline{H})}{\partial \pi(\underline{H})} \frac{\partial \pi(\underline{H})}{\partial P_{\underline{H/A}}} \\
&\quad + \frac{\partial \xi}{\partial e(\underline{D})} \frac{\partial e(\underline{D})}{\partial \pi(\underline{D})} \frac{\partial \pi(\underline{D})}{\partial \pi_D(\underline{H})} \frac{\partial \pi_D(\underline{H})}{\partial \pi(\underline{H})} \frac{\partial \pi(\underline{H})}{\partial P_{\underline{H/A}}} \\
&= -2\pi(\underline{A})[e(\underline{C})^{\mathrm{T}} P^{\mathrm{T}}_{\underline{C/H}} + e(\underline{D})^{\mathrm{T}} P^{\mathrm{T}}_{\underline{D/H}}] \, .
\end{aligned} \tag{49}$$

Note that we do not have to normalise $Bel(\underline{C})$ and $Bel(\underline{D})$ since, providing that $D(\underline{A})$ was normalised before propagation, they are already normalised.

6. The algorithm for adjusting the conditional probabilities in the link matrices

Once we have found the gradients for the link matrices, we can proceed to obtain solutions using the objective function (8). From the equations in Sections 4 and 5, it is obvious that a closed-form solution is very difficult to obtain since it requires us to invert all the matrices. Hence, we choose to obtain the solutions by numerical iteration, using a method similar to the least mean-squared algorithm [34]. We avoided the established Newton–Raphson method [27], since, for each equation, the Hessian of the objective function must be obtained, and if we use the secant approximation method, we need to have two initial values. By comparison, if we use the least mean-squared algorithm, we only need the Jacobian of the objective function and one set of initial values.

Bayesian networks are subject to more constraints than other connectionist networks, such as neural networks, and we will now look at these. First, all the conditional probabilities of the link matrices must be positive definite. In the least mean-squared algorithm, if the learning rate is too large, then there is a danger that some of the elements of the link matrix may become negative. At first sight, we may infer that during training the learning rate must be chosen as small as possible. However, an alternate strategy is to choose a learning rate that is large and impose a check to make sure that the differential matrix will not cause any element of the link matrix to become negative. More loosely, we can allow the elements to become negative and reset their values to zero after each iteration. If the data in the delta matrix is too large, we can scale the whole matrix until the constraint is satisfied. This in effect is to dynamically adjust the learning rate. From experience, this strategy helps the model to converge faster. However, it introduces a further problem which occurs in cases where the gradient is negative, and the corresponding element in the matrix is 0. Any further adjustment to the matrix will not satisfy the axioms of probability, and the process cannot continue even though a minimum has not been reached. To get round this, we add a small constant value perturbation to each of the conditional probabilities and continue with the training algorithm.

The second constraint in a Bayesian network is that all the conditional probabilities must sum to one for any given causal variable. That is

$$\sum_{i=1}^{A} P(b_j/a_i) = 1\,.$$

To satisfy this constraint, we always normalise the conditional probability after each propagation. This added constraint reduces the number of possible solutions, when we compare it with similar connectionist networks. The constraint is not sufficient to ensure that we obtain a unique solution for different initial conditions, but it apparently does not affect the performance.

Given $P(X)$, for the added hidden node we define $P(X, \boldsymbol{H})$ where $\boldsymbol{H}$ is the virtual evidence for the hidden node. Since we do not have any information about the distribution of $\boldsymbol{H}$, we choose a random number for each state of h_i, and normalise the result to 1. Hence, the initial expectation is well-distributed white noise in the hidden node.

The algorithmic steps are summarised as follows:

Creating hidden nodes.

```
Build an initial Bayesian network using subjective assessment or the maximum
  weighted spanning tree algorithm.
Calculate the conditional correlation coefficient by Eq. (4) and/or mutual
  information by Eq. (51) between all nodes in the network.
For each pair of nodes with significant common information:
  Create a hidden node as their unobservable parent.
  Monitor the performance using the testing data set, and train the star structure
    as shown below.
Save the new graph and the conditional probabilities of all the links.
```

Training a star structure.

```
Call Initialisation.
Initialise the learning_rate (the relaxation constant)
While (the mean-squared error improves in each epoch)
  Use Back propagation to obtain the backward gradients.
  Adjust the conditional probabilities based on the gradients using the operating
    equation (5).
  Normalise all the link matrices to satisfy the axioms of probability.
  Use Forward propagation to obtain the forward gradients.
  Adjust the conditional probabilities based on the gradients using the operating
    equation (5).
  Normalise all the link matrices to satisfy the axioms of probabilities.
  If the error does not reduce further due to some elements being 0s.
    Add some perturbation to the whole matrix before the next iteration.
Save the conditional probabilities of the link matrices.
```

Initialisation.

```
Determine the size of the link matrices that satisfy the data dimensions.
  Either: obtain the initial link matrices from expert knowledge
  or: generate random initial values for the hidden variables.
Normalise the conditional probabilities of the link matrices.
```

Back propagation (for one epoch).

```
For each set of training data:
  Assign the data for the leaf nodes as their lambda values.
  Assign the data for the root as the expected result.
  Propagate the lambda messages up to the root node using the operating
    equations (23)–(30).
  Calculate the gradients for each of the conditional probabilities in the link
    matrices using the operating equations (32), (36) and (37).
Sum up all the gradients for each set of training data to obtain the expected
  gradients for each of the link matrices using Eq. (38).
```

Forward propagation (for one epoch).

```
For each set of training data:
  Assign the data for the root nodes as the pi values.
  Assign the data for the leaf nodes as the expected values for the leaf nodes.
  Propagate the pi messages down to the leaf nodes using the operating
    equations (39)–(46).
  Calculate the gradients for each of the conditional probabilities in the link
    matrices using the operating equations (47)–(49).
Sum up all the gradients for each set of training data to obtain the expected
  gradients for each link matrix using Eq. (38).
```

7. Extension to a general singly connected Bayesian network

The gradient equations in Sections 4 and 5 can be easily extended to a star-decomposition with multiple evidence nodes, such as those mentioned in [22].

Fig. 5 shows the general case of a star-decomposable multiple evidence node, where $\boldsymbol{H}$ is the hidden node, and $\lambda(\underline{C_m})$ is the partial belief of the mth child of the hidden node, which takes into account all the effects of its children. Note that the hidden node $\boldsymbol{H}$ has other children as indicated by the ellipsis in Fig. 5.

Most of the operating equations are similar to Section 4, except that the λ value of the hidden node, $\lambda(\underline{H})$, is defined as

$$\lambda(\underline{H}) = \prod_{i=1}^{m} \lambda_{C_i}(\underline{H}) . \tag{50}$$

The Π symbol represents the term-by-term product of all λ messages from the children of the hidden node, i.e., a combination of all the beliefs from all the descendants of the hidden node. The expressions for the gradient $\partial\xi/\partial P_{\underline{H/A}}$ remain the same as in Eq. (32) because the partial derivatives are computed from the root node and chain back to the link between A and the hidden node.

For $\partial\xi/\partial P_{\underline{C_k/H}}$, the terms in the partial derivative are the same, except that

$$\frac{\partial\lambda(\underline{H})}{\partial\lambda_{C_k}(\underline{H})} = \prod_{\substack{i=1\\ i\neq k}}^{m} \lambda_{C_i}(\underline{H}) .$$

The rest of the derivation is obvious. We have also verified that the gradient

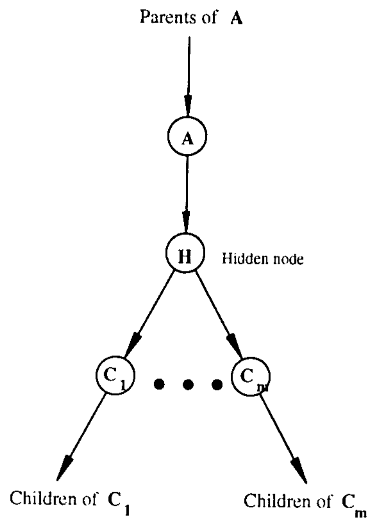

Fig. 5. Extension of Fig. 4 to a general tree.

updating scheme can be easily extended to multiple parents and multiple children networks by simply replacing the corresponding pi message or lambda message with the partial derivative. We will not show the details in this paper as they are not required for our current implementation.

8. Experimental results

In our application, we construct our Bayesian network by acquisition of subjective knowledge from the expert. For example, the sub-module of our system based on the discrete Fourier domain consists of three evidence nodes {[X-size], [Y-size], [Energy]} and two inference nodes {[Advice], [LDR]}. [Advice] is the root node and is the decision node of our system. [LDR] is an intermediate node, which is introduced to make the network more coherent. It serves a similar purpose to the hidden nodes in our current research. There is a distinction in our usage between the terms *intermediate node* and *hidden node*. In the topology of the directed acyclic graph, they are identical and serve the same purpose. However, when we call a node *intermediate*, we mean that its probability parameters are obtained from an expert by subjective means. When we call a node *hidden*, we mean that it is transparent only during data collection and the expert is unaware of its existence. Its probability parameters are found from the world of feasible values which satisfy some loss criteria. In our approach, the identifiable features themselves can be virtual, which means they can be the posterior belief of a probabilistic network.

Without investigating the dependency of the data, we derived the inference diagram shown in Fig. 6 from subjective intuition.

The [LDR] node, which represents a qualitative interpretation of the image, independent of the detection of the lumen, is caused by the amount of room available to advance the endoscope tip. There can be other causes for it, such as temporal data represented by the state of the node at the previous time step. However, in the closed world approach, we do not include all possible causes to the variables in order to make the problem manageable. From the inference

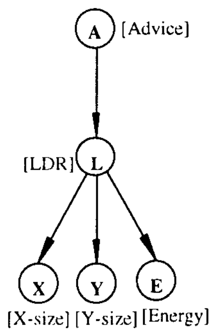

Fig. 6. Inference diagram for the Fourier domain sub-module.

diagram, if we have an interpretation of [LDR] from expert knowledge, the causes to [LDR] (i.e., the value of [Advice]) will not change the likelihood from the node [LDR] to its children. We say that the node [LDR] d-separates {[X-size], [Y-size], [Energy]} from their other predecessors [23]. Making [LDR] the direct cause of the other variables [X-size], [Y-size] and [Energy] looks reasonable. If there is an LDR in the colon image, then it is likely to have a characteristic size detected in the x and y directions and the overall image intensity will be low, thus creating a characteristic energy measured in the image.

However, with some further subjective assessment of the structure, we observed that there are cases where there is no visible large dark region that can be detected as lumen, but the overall colon image is dark. These types of images occur where the colon bends and we can move the endoscope forward. Hence we modified the structure to Fig. 3, where the parent of [Energy] is the proposition variable [Advice] instead of [LDR].

We have chosen to construct the tree initially from subjective data since we think this will reduce the possibility of spurious dependencies in the network. However, other approaches are possible. Cooper and Herskovits [2] suggested an alternative Bayesian method which searches through the possible structures for a network which best satisfies the data.

Chow and Liu [1] have shown that to maximise the maximum likelihood estimate of independent observations $(\boldsymbol{X}^1, \boldsymbol{X}^2, \boldsymbol{X}^3, \ldots, \boldsymbol{X}^K) \in \boldsymbol{X}$ is equivalent to maximising the mutual information between each link. The network which does this is known as the maximum weighted spanning tree. In probabilistic terms, mutual information [5] defined as

$$I(a_i, b_j) = \log \frac{P(a_i/b_j)}{P(a_i)} = \log \frac{P(a_i, b_j)}{P(a_i)P(b_j)} .$$

Summing over all the states for variables $\boldsymbol{A}$ and $\boldsymbol{B}$, weighted by their joint probability, we have

$$I(\boldsymbol{A}, \boldsymbol{B}) = \sum_{B} \sum_{A} P(\boldsymbol{A}, \boldsymbol{B}) \log \frac{P(\boldsymbol{A}, \boldsymbol{B})}{P(\boldsymbol{A})P(\boldsymbol{B})} \tag{51}$$

which is a measure of the information provided about the event $\boldsymbol{A}$ given the occurrence of event $\boldsymbol{B}$. The likelihood of $\boldsymbol{X}$ is defined as the product of individual probabilities,

$$L(\boldsymbol{X}^1, \boldsymbol{X}^2, \boldsymbol{X}^3, \ldots, \boldsymbol{X}^K) = \prod_{k=1}^{p} P(\boldsymbol{X}^k) . \tag{52}$$

In our application, our observations may not be statistically independent. Hence the correct likelihood definition for our observations should be

$$L(\boldsymbol{X}^1, \boldsymbol{X}^2, \boldsymbol{X}^3, \ldots, \boldsymbol{X}^K) = \prod_{k=1}^{p} P(\boldsymbol{X}^k / \boldsymbol{X}^1 \cdots \boldsymbol{X}^{k-1}) . \tag{53}$$

We seek to strike a balance between the subjective and objective approach in

Table 1
Mutual information for {[LDR], [X-size], [Y-size]}

Mutual information between		Result
LDR	X-size	0.1863
LDR	Y-size	0.1900
X-size	Y-size	1.2276

constructing our inference diagram. After initial subjective construction, we use statistical analysis to find the correlation and mutual information for the data in each branch. If there are variables with the same cause that are highly correlated and have high mutual information, then it is most likely that we can improve the network performance by the creation of hidden nodes to remove the over-emphasis of their common information.

The mutual information for the variables {[LDR], [X-size], [Y-size]} is summarised in Table 1. If we were to use the maximum weighted spanning tree algorithm [1, 6] to build a tree from this information we would obtain the structure of Fig. 7.

Since we have a measurement for Y-size in our application, we treat this as virtual evidence for the node [Y-size], denoted as an additional child node, which propagates a lambda message to the node [Y-size].

The tree obtained using our hidden node approach is shown in Fig. 8.

When we evaluated the performance of the maximum weighted spanning tree for the estimate of [LDR] in Fig. 7 we found that the performance, measured by the probability of a correct deduction, improved from 0.6139 to 0.6436 compared to the original subjective structure. However, when we used the hidden node structure as in Fig. 8, using 100 sets of training data and 202 test sets, following

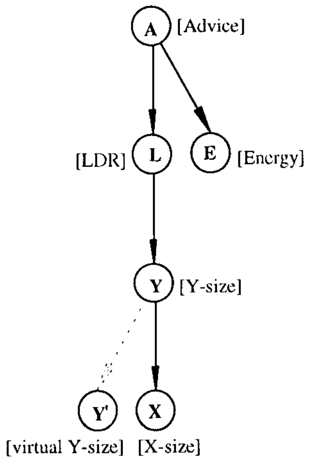

Fig. 7. The inference diagram by MWST.

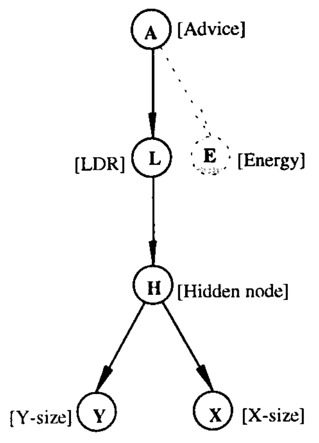

Fig. 8. The inference diagram with hidden node.

the training algorithms outlined in the previous sections, we found a much greater improvement to 0.84, for the detection of [LDR].

Fig. 9 summarises some of the statistics obtained during training. Fig. 9(a) shows the sum of squared-error for the prediction of [LDR] given [X-size] and [Y-size], after each iteration of back propagation and forward propagation. The epoch axis shows the number of iterations. Notice that our initial guess is not too bad, because we are using the actual statistical data $P(\text{LDR, X-size, Y-size})$ with a random number initial estimate of the joint probability of $P(\text{LDR, X-size, Y-size, Hidden})$. Fig. 9(b) shows the sum of squared-error for the prediction of [X-size]

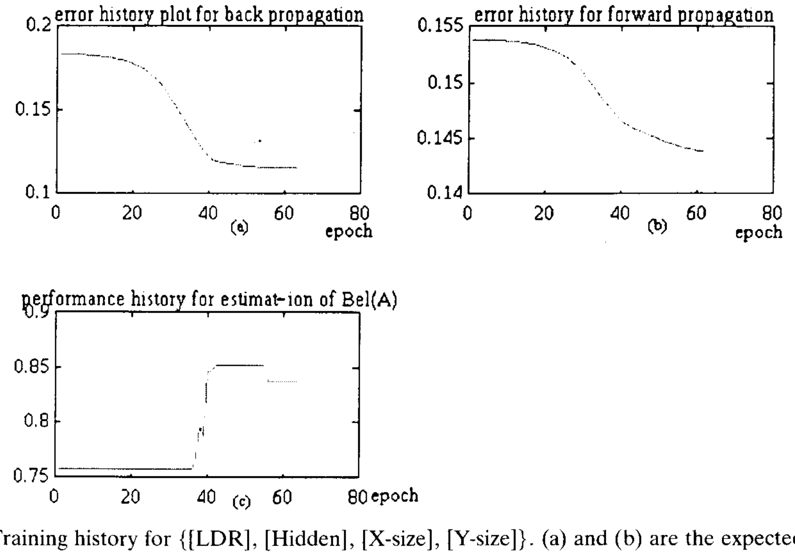

Fig. 9. Training history for {[LDR], [Hidden], [X-size], [Y-size]}. (a) and (b) are the expected sum of squared-errors during training and (c) is the performance of the test data set.

and [Y-size] given [LDR]. Fig. 9(c) shows the performance of the prediction of [LDR]. Notice that there is a small drop in the performance. This is due to the fact that many of the posterior probabilities are near the decision threshold. Our strategy is to minimise the sum of squared-error, which associates a high penalty cost for those cases where the predictions are at the opposite extreme. This phenomenon only occurs in the training of the Fourier domain sub-module. When we tested it with the other modules, the minimised sum of squares algorithm gives the best performance. In general, the minimised sum of squared-error is higher than in a corresponding neural network. This is due to the absence of a squashing function in the Bayesian network.

Fig. 10 plots the conditional probabilities of the link matrices for (a) [LDR] to the hidden node [H], (b) the hidden node [H] to [X-size], (c) the hidden node [H] to [Y-size].

After training the first hidden node, and carrying out further statistical tests, we introduced a second hidden node separating [LDR] and [Energy] from [Advice]. The new configuration of our Bayesian network is shown in Fig. 11. Notice that the sub-network is a single chain, if we express the lambda and pi messages for [G] and [H], we have

$$\begin{aligned} \lambda(\boldsymbol{G}) &= \lambda_L(\boldsymbol{G}) = P_{\underline{L/G}}\lambda(\boldsymbol{L}) = P_{\underline{L/G}}P_{\underline{H/L}}\lambda(\boldsymbol{H})\,, \\ \pi(\boldsymbol{H}) &= P^{\mathrm{T}}_{\underline{H/L}}\pi(\boldsymbol{L}) = P^{\mathrm{T}}_{\underline{H/L}}P^{\mathrm{T}}_{\underline{L/G}}\pi(\boldsymbol{G}) = [P_{\underline{L/G}}P_{\underline{H/L}}]^{\mathrm{T}}\pi(\boldsymbol{G})\,. \end{aligned} \tag{54}$$

If we define $P_{\underline{H/G}} = P_{\underline{L/G}}P_{\underline{H/L}}$, we see that we can remove the internal node [LDR] in our advisory system since we never needed to know the belief of [LDR] in the application. By adjusting the weights of the links to hidden node [G] we can combine the contribution of [LDR] into [G]. Thus the structure of Fig. 11 can be simplified to the structure shown in Fig. 12.

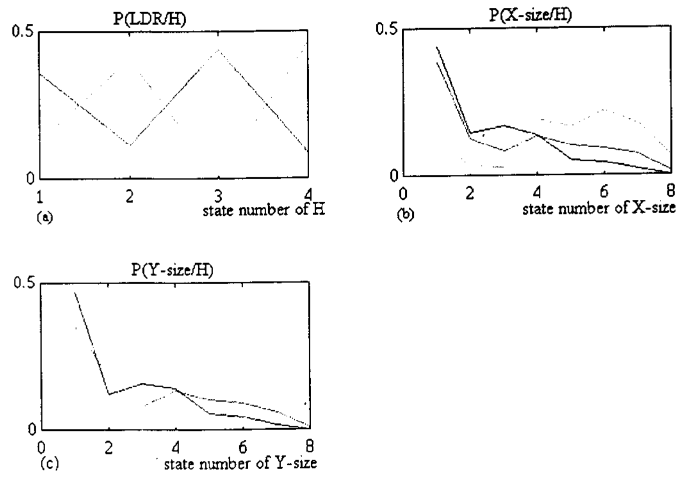

Fig. 10. The conditional probabilities of the link matrices.

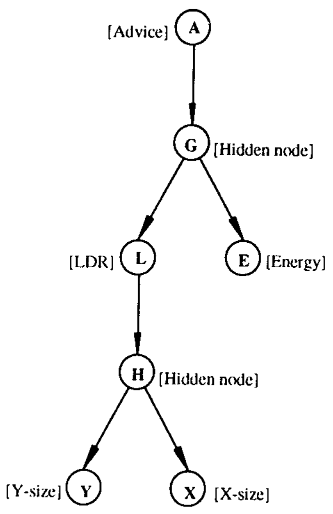

Fig. 11. Fourier domain sub-module with second hidden node *G*.

Fig. 13 summarises some of the statistics during training. Fig. 13(a) shows the sum of squared-error for the prediction of [Advice] given [X-size], [Y-size] and [Energy], after each iteration of back propagation and forward propagation. Fig. 13(b) shows the sum of squared-error in the forward prediction of [H] and [Energy] given [Advice]. Fig. 13(c) shows the performance of the prediction of [Advice] in the testing data set. A small perturbation at iteration 164 was added to all conditional probabilities when the system reached a minimum point. This approach is similar to the well-known relaxation techniques employed in the Boltzmann machine [10]. We added the perturbation to allow the system to search for an alternative minimum point which may better model the training data. Although there was no improvement in the number of correct classifications in the

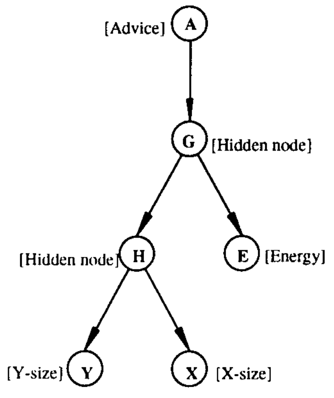

Fig. 12. Fourier domain sub-module with hidden nodes *G, H*.

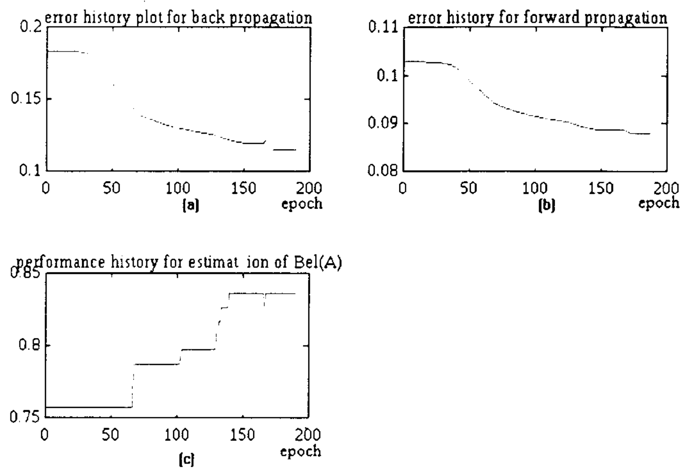

Fig. 13. Training history for {[Advice], [*G*], [*H*], [Energy]}. (a) and (b) are the expected sum of squared-errors during training and (c) is the performance of the test data set.

test data, the lower sum of squared-errors implies that we managed to find a better fit than the first minimum.

Table 2 summarises the results of the different configurations of the Fourier domain sub-modules shown in Figs. 6, 7 and 12.

From Fig. 8, if we ignore the estimation of intermediate node [LDR] and define the link matrix $\underline{P_{H/A}} = \underline{P_{L/A}}\,\underline{P_{H/L}}$, we can train the hidden node [*H*] with only observed data.

We tried the same approach on a second sub-module of our system which uses region-based segmentation to estimate the lumen [12]. The network consisted of five nodes {[Advice], [LDR], [Size], [Mean], [Variance]} and was used experimentally by Sucar and Gillies [32] to investigate the effect of correlated data in Bayesian networks. Sucar investigated the configurations for the model as shown in Fig. 14.

Fig. 14(a) is the original model derived from expert knowledge, Fig. 14(b) is a modified structure with the [Mean] node deleted, since it was found to be correlated with the [Variance] node. The unconditional correlation coefficient, originally employed by Sucar [30] to test dependency, is 0.4947. However, since we are investigating the conditional independence assumption, it is more correct to use the conditional correlation coefficient of Eq. (4). The conditional correlation over [LDR] is 0.5880.

Table 2
Performance of various configurations for Fourier domain sub-module

Configuration	Performance
Fig. 6 (direct)	0.6733
Fig. 7 (MWST)	0.6832
Fig. 12 (hidden node)	0.8366

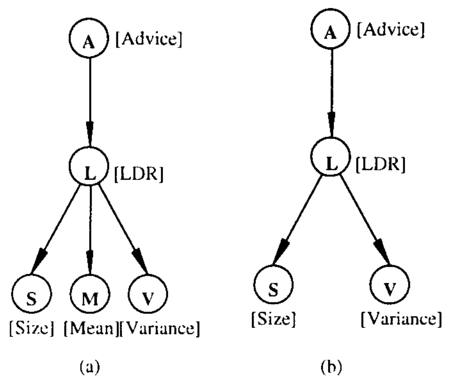

Fig. 14. Configurations of the region-based segmentation model.

The mutual information for the variables {[LDR], [Size], [Mean], [Variance]} is summarised in Table 3, which further supports the claim that there is information within the [Mean] and [Variance] nodes that should be neutralised.

We combined the contribution from [Mean] and [Variance] using a first hidden node [*H*] (Fig. 15(a)), ignoring the internal node [LDR], and then with [Size] linked through the second hidden node [*G*] (Fig. 15(b)).

Fig. 16 shows some statistics during the training of the hidden node [*H*]. Fig. 16(a) shows the history of the sum of squared-error for the back propagation prediction of [Advice], Fig. 16(b) shows the history of the sum of squared-error for the forward propagation prediction of {[Mean], [Variance]}, and Fig. 16(c) shows the performance of the prediction of [Advice]. After about forty iterations it was found that the process was not converging due to some of the probabilities in the link matrices falling to zero. Hence a perturbation of 0.2 was added to all conditional probabilities, and the convergence continued as shown in Fig. 17.

The program terminates when further adjustment of the link matrices does not improve the sum of squared-error. Note the increase in the sum of squared-errors and the deterioration of the prediction performance immediately after the perturbation.

Fig. 18 is a snapshot of the link matrix between [*H*] and [Mean], Fig. 18(a) is the initial estimate, Fig. 18(b) shows the state after 25 iterations, Fig. 18(c) after

Table 3
Mutual information for {[LDR], [Size], [Mean], [Variance]}

Mutual information between		Result
LDR	Size	0.2827
LDR	Mean	0.2223
LDR	Variance	0.1149
Size	Mean	0.4098
Size	Variance	0.2999
Mean	Variance	0.4841

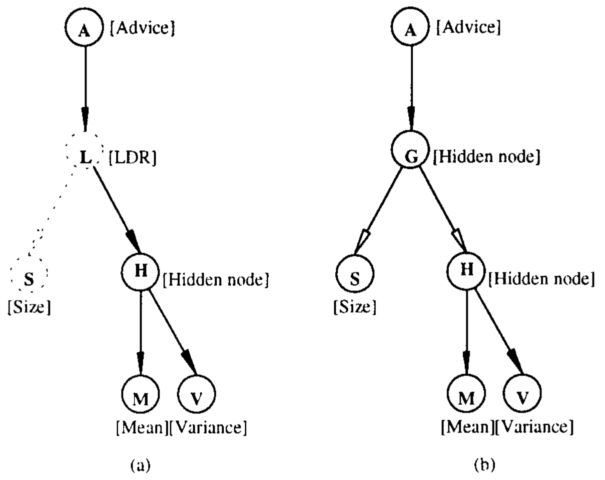

Fig. 15. Hidden node creation for the region-based segmentation model.

110 iterations and Fig. 18(d) the total changes to the conditional probabilities of the link matrix.

Fig. 19 shows statistics collected during the training of the hidden node $[G]$. In Fig. 15(b) a perturbation of 0.5 was added to all the link matrices to remove the zero elements. Fig. 19(a) shows the history of the sum of squared-error for the forward propagation prediction of {[Size], $[H]$}, and Fig. 19(c) the performance of the prediction of [Advice] which finishes at 0.9010.

In creating the nodes $[H]$ and $[G]$, we ignored the requirement to collect data

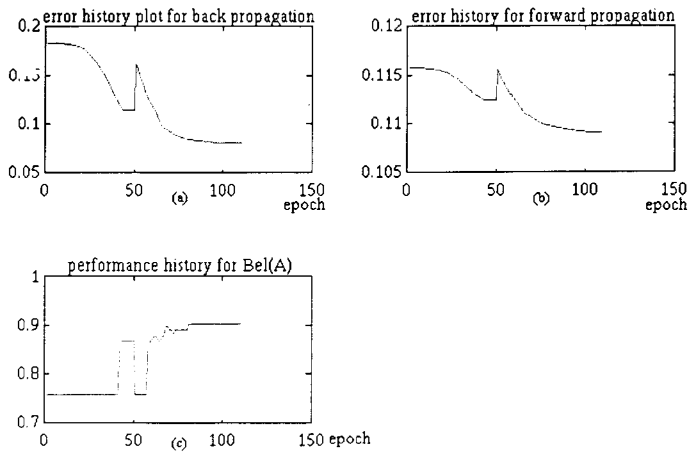

Fig. 16. Training of hidden node $[H]$ for the region-based segmentation model. (a) and (b) are the expected sum of squared-errors during training and (c) is the performance of the test data set.

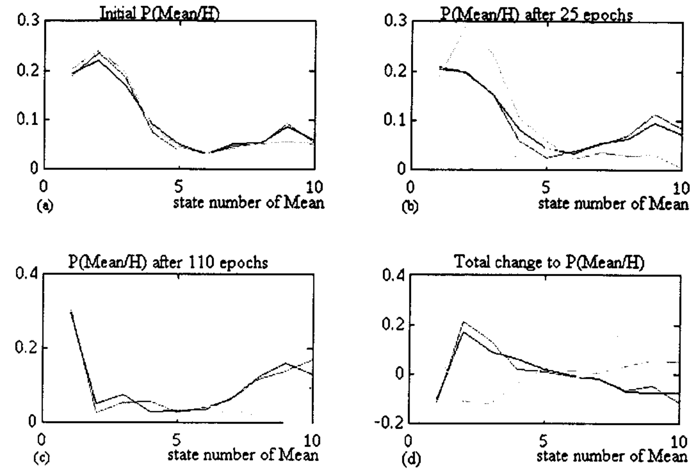

Fig. 17. Training history of hidden node [H] after perturbation. (a) and (b) are the expected sum of squared-errors during training and (c) is the performance of the test data set.

for the [LDR] node in our model, which, as noted previously is an internal node whose value is not required. In effect, Figs. 15(a) and 15(b) represent the same structure, with the exception that [G] has more states than [LDR]. The predictive performance turns out to be the same, which gives further justification to the validity of the training algorithm.

Fig. 20 depicts the link matrices between the nodes: Fig. 20(a) shows [Advice] to [G]; Fig. 20(b) [G] to [Size] and Fig. 20(c) [G] to [Hidden]. We notice that a characteristic for the hidden nodes is that the link matrices exhibit little prediction

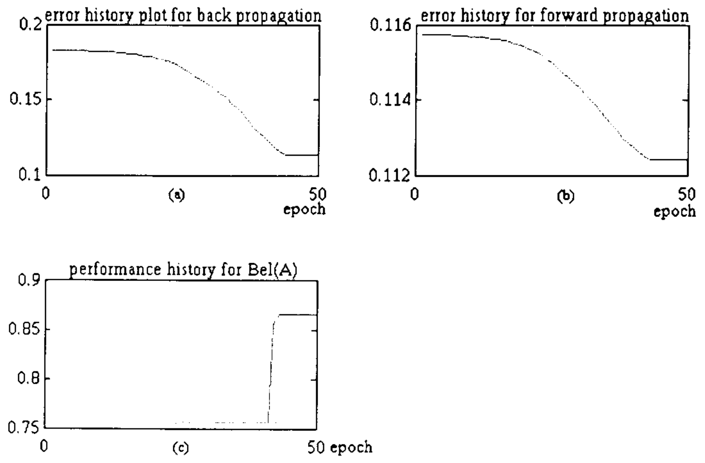

Fig. 18. Snapshot of the link matrix between [H] and [Mean].

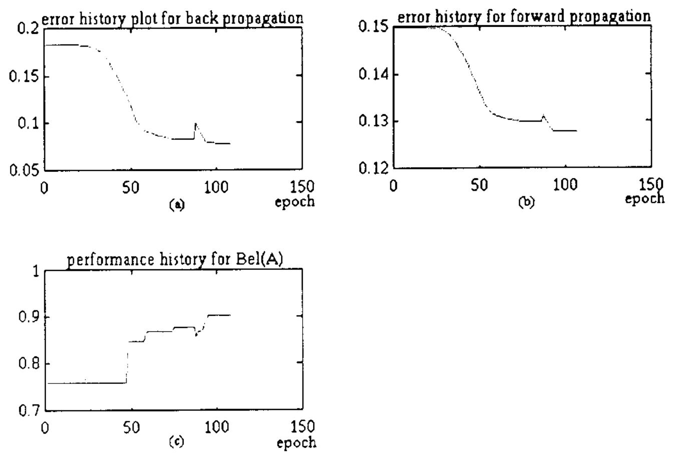

Fig. 19. Creation of hidden node [G] for the region-based segmentation model. (a) and (b) are the expected sum of squared-errors during training and (c) is the performance of the test data set.

of prior probability. From the link matrices in Figs. 20(a) and 20(c), the sum of all the conditional probabilities is almost a constant value, which gives no estimate of the prior probability. However, summing the conditional probabilities for each value of [Size] and normalising the result to satisfy the axioms of probability, we can obtain a close estimate of the prior probability of [Size] from the training data.

Table 4 summarises the result of the different configurations of the region-based segmentation model as shown in Figs. 14 and 15(b).

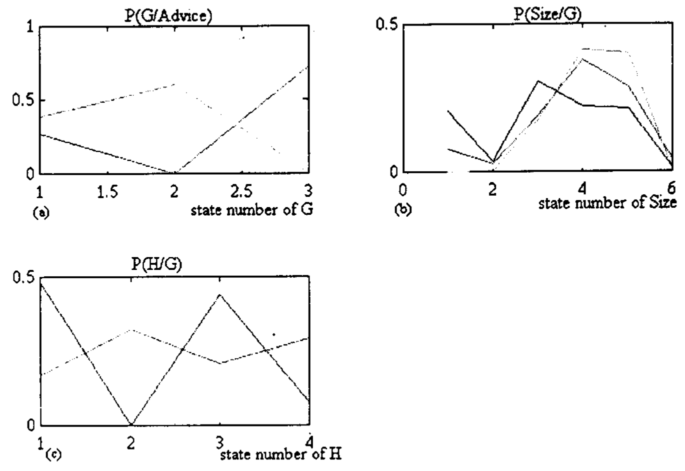

Fig. 20. The link matrices after the creation of node [G].

Table 4
Performance of various configurations for the region-based segmentation model

Configuration	Performance
Fig. 14(a) (direct)	0.7376
Fig. 14(b) (node deletion)	0.7475
Fig. 15(b) (hidden node)	0.9010

9. Conclusions and suggestions for future work

From the experimental results it is clear that the creation of the hidden node as an unobservable variable can be a powerful tool to handle conditionally dependent data in Bayesian networks. By creating hidden nodes, and training them using Bayesian operating equations according to the criterion of minimising the squared-error cost function, we have improved the performance of two different networks. Our method utilises the objective probabilities of the training data, constrained by the axioms of probability, and performance is maximised without expert intervention during training of internal nodes. We have also demonstrated that there is no additional data to be collected for the creation of hidden nodes.

At present we create the hidden nodes sequentially starting from the most dependent data. Future research should be undertaken to compare this approach with direct minimisation of the cost function with all hidden nodes created before training. Further investigation is also required to determine how many states are necessary for the hidden nodes, how to select a dynamic learning rate, how to estimate initial values for the hidden variable(s) and how to trim the size of the network when a column and row of conditional probabilities fall to zero during training.

Appendix A

If the variable $\boldsymbol{A}$ has A elements and the training data $\boldsymbol{E} \in \Im$, the full set of training data, and we are interested in the posterior probabilities (belief) of $\boldsymbol{A}$ given some evidence $\boldsymbol{E}$, denoted as $Bel(\boldsymbol{A})$, then we express $Bel(\boldsymbol{A})$ as a function of all the evidence

$$Bel(\boldsymbol{A}) = f(\boldsymbol{E}) .$$

If we have the expectation of the squared error cost function, Δ, as

$$\xi = \sum_{i=1}^{A} [D(a_i) - Bel(a_i)]^2 ,$$

$$\Delta = E\{\xi\} = E\left\{\sum_{i=1}^{A} [D(a_i) - Bel(a_i)]^2\right\} ,$$

then, following the method used by Hampshire and Pearlmutter [8], using the joint probability density function of $P(\boldsymbol{E}, D(\boldsymbol{A}))$ of the training data and the desired output, we have

$$\begin{aligned}\Delta &= E\left\{\sum_{i=1}^{A}[D(a_i) - Bel(a_i)]^2\right\}\\ &= \sum_{\boldsymbol{E}\in\Im}\sum_{a_j\in A}\left\{\sum_{i=1}^{A}[D(a_i) - Bel(a_i)]^2\right\}P(\boldsymbol{E}, a_j)\,.\end{aligned}$$

The second summation is applied to each element of the desired value, so we can change the representation to

$$\begin{aligned}\Delta &= \sum_{\boldsymbol{E}\in\Im}\sum_{j=1}^{A}\left\{\sum_{i=1}^{A}[D(a_i) - Bel(a_i)]^2\right\}P(\boldsymbol{E}, a_j)\\ &= \sum_{\boldsymbol{E}\in\Im}\sum_{j=1}^{A}\left\{\sum_{i=1}^{A}[D(a_i) - Bel(a_i)]^2\right\}P(a_j/\boldsymbol{E})P(\boldsymbol{E})\\ &= E\left\{\sum_{j=1}^{A}\left\{\sum_{i=1}^{A}[D(a_i) - Bel(a_i)]^2\right\}P(a_j/\boldsymbol{E})\right\}\end{aligned}$$

which is the expectation of the squared-error average over the likelihood of each state of the desired output given the evidence. Expanding the squared-error expression we get

$$\begin{aligned}\Delta &= E\left\{\sum_{j=1}^{A}\sum_{i=1}^{A}[(D(a_i))^2 + (Bel(a_i))^2 - 2D(a_i)Bel(a_i)]P(a_j/\boldsymbol{E})\right\}\\ &= E\left\{\sum_{i=1}^{A}\left[\sum_{j=1}^{A}(D(a_i))^2P(a_j/\boldsymbol{E}) + \sum_{j=1}^{A}(Bel(a_i))^2P(a_j/\boldsymbol{E})\right.\right.\\ &\qquad\left.\left. - 2\sum_{j=1}^{A}D(a_i)Bel(a_i)P(a_j/\boldsymbol{E})\right]\right\}.\end{aligned}$$

$Bel(a_i)$ only depends on the evidence and hence it can be moved outside the summation, and further $\sum_{j=1}^{A} P(a_j/\boldsymbol{E}) = 1$, so the equation can be simplified to

$$\begin{aligned}\Delta &= E\left\{\sum_{i=1}^{A}\left[\sum_{j=1}^{A}(D(a_i))^2P(a_j/\boldsymbol{E}) + (Bel(a_i))^2 - 2Bel(a_i)\sum_{j=1}^{A}D(a_i)P(a_j/\boldsymbol{E})\right]\right\}\\ &= E\left\{\sum_{i=1}^{A}[E((D(a_i))^2/\boldsymbol{E}) + (Bel(a_i))^2 - 2Bel(a_i)E(D(a_i)/\boldsymbol{E})]\right\}.\end{aligned}$$

Now,

$$\sum_{j=1}^{A}(D(a_i))^2P(a_j/\boldsymbol{E}) = E\{(D(a_i))^2/\boldsymbol{E}\}$$

and

$$\sum_{j=1}^{A} D(a_i)P(a_j/\boldsymbol{E}) = E\{D(a_i)/\boldsymbol{E}\}$$

are the conditional expectations of $(D(a_i))^2$ and $D(a_i)$ over specific evidence $\boldsymbol{E}$. Adding and subtracting a term

$$\sum_{i=1}^{A} E^2\{D(a_i)/\boldsymbol{E}\}$$

and using the identity

$$\operatorname{var}\{D(a_i)/\boldsymbol{E}\} = E\{(D(a_i))^2/\boldsymbol{E}\} - E^2\{D(a_i)/\boldsymbol{E}\}\ ,$$

we have

$$\begin{aligned}\Delta = E\Bigg\{\sum_{i=1}^{A} [&E^2\{D(a_i)/\boldsymbol{E}\} + (Bel(a_i))^2 - 2Bel(a_i)E\{D(a_i)/\boldsymbol{E}\}\\ &+ E\{(D(a_i))^2/\boldsymbol{E}\} - E^2\{D(a_i)/\boldsymbol{E}\}]\Bigg\}\\ = E\Bigg\{\sum_{i=1}^{A} [&E\{D(a_i)/\boldsymbol{E}\} - (Bel(a_i))]^2\Bigg\} + E\{\operatorname{var}\{D(a_i)/\boldsymbol{E}\}\}\ .\end{aligned}$$

Since the second term is independent of the network outputs, minimisation of the mean-squared-error cost function will be

$$\min\{\Delta\} = \min\left\{E\left\{\sum_{i=1}^{A} [E\{D(a_i)/\boldsymbol{E}\} - (Bel(a_i))]^2\right\}\right\}$$

and $E\{D(a_i)/\boldsymbol{E}\}$ is the expected belief of $\boldsymbol{A}$ given the evidence $\boldsymbol{E}$.

References

[1] C.K. Chow and C.N. Liu, Approximating discrete probability distributions with dependence trees, *IEEE Trans. Inform. Theory* **14** (1986) 462–467.

[2] G.F. Cooper and E. Herskovits, A Bayesian method for constructing Bayesian belief networks for databases, in: B.D. D'Ambrosio, P. Smets and P.P. Bonissone, eds., *Proceedings 7th International Conference on Uncertainty in Artificial Intelligence*, Los Angeles, CA (Morgan Kaufmann, Los Altos, CA 1991) 86–94.

[3] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. Roy. Stat. Soc. Ser. B* **39** (1977) 1–38.

[4] J.S. Denker and Y. Le Cun, Transforming neural-net output levels to probability distributions, in: R.P. Lippmann, J.E. Moody and D.S. Touretzky, eds., *Advances in Neural Information Processing Systems* **3** (Morgan Kaufmann, Los Altos, CA, 1991).

[5] R.G. Gallager, *Information Theory and Reliable Communication* (Wiley, New York, 1973).

[6] D. Geiger, An entropy-based learning algorithm of Bayesian conditional trees, in: D. Dubois, M.P. Wellman, B.D. D'Ambrosio and P. Smets, eds., *Proceedings 8th Conference on Uncertainty in Artificial Intelligence*, Stanford, CA (Morgan Kaufmann, Los Altos, CA, 1992) 92–97.

[7] D. Geiger, T. Verma and J. Pearl, Identifying independence in Bayesian networks, *Networks* **20** (1990) 507–534.

[8] J.B. Hampshire and B. Pearlmutter, Equivalence proofs for multi-layer perceptron classifiers and the Bayesian discriminant function, in: D.S. Touretzky, J.L. Elman, T.J. Sejnowski and G.E. Hinton, eds., *Connectionist Models, Proceedings of the 1990 Summer School* (1990).

[9] D. Heckerman, Probabilistic similarity networks, ACM Doctoral Dissertation Award 1990, MIT, Cambridge, MA (1990).

[10] G.E. Hinton, T.J. Sejnowski and D.H. Ackley, Boltzmann machine: constraint satisfaction networks that learn, Tech. Rept. CMU-CS-84-119 (1984).

[11] G.N. Khan, Machine vision for endoscope control and navigation, Ph.D. Thesis, Imperial College, London (1989).

[12] G.N. Khan and D.F. Gillies, A highly parallel shade image segmentation method, in: P.M. Dew and R.A. Earnshaw, eds., *Parallel Processing for Computer Vision and Display* (Addison-Wesley, Reading, MA, 1989) 180–189.

[13] H.H. Ku and S. Kullback, Approximating discrete probability distribution, *IEEE Trans. Inform. Theory* **15** (1969) 444–447.

[14] C.K. Kwoh and D. Gillies, Using Fourier information for the detection of the lumen in endoscope images, in: *IEEE Region 10's Ninth Annual International Conference, Proceeding of TENCON Conference*, Singapore (1994) 981–985.

[15] H.J. Larson, *Introduction to Probability Theory and Statistical Inference* (Wiley, New York, 3rd ed., 1982).

[16] S.L. Lauritzen, The EM algorithm for graphical association models with missing data, *Comput. Stat. Data Anal.*, **19** (1995) 191–201.

[17] S.L. Lauritzen and D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, *J. Roy. Stat. Soc. Ser. B* **50** (1988) 157–224.

[18] Y. Le Cun, J.S. Denker and S.A. Solla, Optimal brain damage, in: D.S. Touretzky, ed., *Advances in Neural Information Processing Systems* **2** (Morgan Kaufmann, Los Altos, CA, 1990).

[19] R.M. Neal, Connectionist learning of belief networks, *Artif. Intell.* **65** (1991) 71–133.

[20] R.E. Neapolitan, *Probabilisitc Reasoning in Expert Systems: Theory and Algorithms* (Wiley, New York, 1990).

[21] G. Paass, Integrating probabilistic rules into neural networks: a stochastic EM learning algorithm, in: B.D. D'Ambrosio, P. Smets and P.P. Bonissone, eds., *Proceedings 7th International Conference on Uncertainty in Artificial Intelligence*, Los Angeles, CA (Morgan Kaufmann, Los Altos, CA, 1991) 264–270.

[22] J. Pearl, Fusion, propagation, and structuring in belief networks, *Artif. Intell.* **29** (1986) 241–288.

[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, Los Altos, CA, 1988).

[24] F.P. Ramsy, Truth and probability (1926), reprinted in: H.E. Kyburg and H.E. Smokler, eds., *Studies in Subjective Probability* (Wiley, New York, 1964) 63–92.

[25] H. Rashid and P. Burger, Differential algorithm for the determination of shape from shading using a point light source, *Image Vision Comput.* **10** (1992) 119–127.

[26] M.D. Richard and R.P. Lippmann, Neural network classifiers estimate Bayesian a posteriori probabilities, *Neural Comput.* **3** (1990) 461–483.

[27] S.M. Roberts and J.S. Shipman, *Two-Point Boundary Value Problems: Shooting Methods* (Elsevier, Amsterdam, 1972) Chapter 6.

[28] S.A. Solla, E. Levin and M. Fleisher, Accelerated learning in layered neural networks, *Complex Syst.* **2** (1988) 625–640.

[29] D.J. Spiegelhalter, A.P. David, S.L. Lauritzen and R.G. Cowell, Bayesian analysis in expert systems, *Stat. Sci.* **8** (1993) 219–283.

[30] L.E. Sucar, Probabilistic reasoning in knowledge-based vision systems, Ph.D. Thesis, Imperial College, London (1992).

[31] L.E. Sucar and D.F. Gillies, Probabilistic reasoning in high-level vision, *Image Vision Comput.* **12** (1994) 42–60.

[32] L.E. Sucar, D.F. Gillies and D.A. Gillies, Objective probabilities in expert systems, *Artif. Intell.* **61** (1993) 187–208.

[33] A.S. Weigend, D.E. Rumelhart and B.A. Huberman, Back-propagation, weight-elimination and time series prediction, in: D.S. Touretzky, J.L. Elman, T.J. Sejnowski and G.E. Hinton, eds., *Connectionist Models, Proceedings of the 1990 Summer School* (1990).

[34] B. Widrow and S.D. Stearns, *Adaptive Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1985).

[35] L. Xu and J. Pearl, Structuring causal tree models with continuous variables, in: L.N. Kanal, T.S. Levitt and J.F. Lemmer, eds., *Uncertainty in Artificial Intelligence* **3** (1989) 209–219.