5

Probabilistic Inference for Hybrid Bayesian Networks

Wei Sun and Kuo-Chu Chang
Department of Systems Engineering & Operations Research
George Mason University, Fairfax, VA
USA

1. Introduction

A Bayesian network (BN) Charniak (1991) Pearl (1988) Jensen (1996) Neapolitan (1990) is a directed acyclic graph (DAG) consisting of nodes and arrows, in which node represents random variables, and arrow represents dependence relationship between connected nodes in the sense of the probabilistic, deterministic, or functional. Each node in BN has a specified conditional probability distribution (CPD), where all CPDs together parameterize the model. BNs have been used as powerful probabilistic knowledge models for decision support under uncertainty over a few decades, with numerous applications such as classification, medical diagnosis, bioinformatics, speech recognition, etc. One of the most important features BN has is the factorization of the joint probability space, so that conditional independence can be exploited to simplify modeling and save computations. However, BN model is only useful when combined with efficient algorithms for inference.

Over decades after the first Bayesian network (BN) was introduced in early 1980s, a number of inference algorithms have been reported in the literature. However, for hybrid Bayesian networks with both discrete and continuous variables, which are usually inevitable in modeling real-life problems, inference task has many difficulties and open issues. This chapter focuses on introducing the state-of-the-art hybrid inference methods in the literature. Particularly, we take scalability as a very important aspect and intend to provide the reader the opportunities to get the efficent inference methods under different circumstances.

The simplest hybrid Bayesian network is called Conditional Linear Gaussian (CLG) and it is a hybrid model for which exact inference can be performed by the Junction Tree (JT) algorithm Lauritzen (1992). However, JT and all of other exact inference algorithms have the complexity of being, in general, exponential to the size of the largest clique of the strongly triangulated graph. For a hybird BN model, there surely exists hybrid cliques that including all of discrete parent nodes for a connected continuous subgraph and at least one continuous node from the subgraph, which is usually the largest clique. Therefore, in most of real applications, exact inference is intractable.

For a general hybrid Bayesnet, due to the difficult issues such as the heterogeneity of variables, arbitrary densities involved, and possibly any functional relationships, with network topologies that may have discrete variables as parents of continuous nodes, we have

to rely on approximate inference with tradeoff in accuracy against complexity. To this end, there are several main categories of approximate algorithms:

1. Generalized Junction Tree algorithm: when the complexity of a network is beyond the capability of traditional JT, Koller et al. (1999) proposed a general algorithm under the framework of Junction Tree, but using approximate clique potentials to do the clique tree propagation. It involves approximate inference algorithms to estimate the densities in each clique. Further, for hybrid network with the structure such that continuous variable has discrete children, Shenoy (2006) introduces a way to convert the model into a network with CLG structure, and then use Gaussian mixtures to approximate clique potentials for inference under Junction Tree framework. Another interesting method to approximate densities is to use truncated exponential Cobb & Shenoy (2006).
2. Hybrid loopy belief propagation: also know as message passing, the first belief propagation algorithm was proposed by Pearl in 1980s Pearl (1988), to provide exact inference for discrete polytree BN model. When there is any loop in the network, it becomes loopy belief propagation and provides accurate approximate solutions empirically for discrete networks Murphy et al. (1999). In hybrid case, Yuan & Druzdzel (2006) proposed a computationally extensive approach combining nonparametric belief propagation Sudderth et al. (2003), numerical integration, and density estimation techniques to pass messages between any types of variables.
3. Monte Carlo: importance sampling methods, such as Likelihood Weighting Fung & Chang (1989), Shachter & Peot (1999), are model-free algorithms, but usually have difficulty in dealing with unlikely evidence. The state-of-the-art importance sampling algorithms are AIS-BN Cheng & Druzdzel (2000) and EPIS-BN Yuan & Druzdzel (2007). Unfortunately, both work for discrete networks only. For hybrid BN models, any approximate results obtained by algorithms in the first two categories can be certainly used as the importance functions for efficient sampling process. Other sampling methods include Markov Chain Monte Carlo (MCMC) Gilks et al. (1996), Gamerman & Lopes (2006).
4. Variational methods: by formulating probabilistic inference into an optimization problem, variational methods provide another perspective for approximation solutions (Wainwright & Jordan (2008)).

We are particularly interested in the message passing framework because of its simplicity of implementation and good empirical performance, and more importantly, its distributed nature of inference. Without the computational burden of numerical integration, we proposed a partitioned message passing algorithm in Sun & Chang (2009), using interface nodes to separate the original network into sub-networks. Each sub-network contains only one type of variables, either discrete or continuous. We then conduct message passing separately within each sub-network. Finally, messages are fused together through interface nodes and the posterior distributions are computed based on final messages. The advantage of the partitioned message passing method is that it is easier to accommodate an efficient algorithm for inference within homogeneous sub-networks. On the other hand, a disadvantage is that we have to conduct inference conditioning on all the discrete parent nodes (i.e., interface nodes), for each connected continuous subgraph. Therefore, the algorithm has an exponential complexity proportional to the product of sizes of discrete parent nodes.

It is more desirable to have an unified message passing framework that allows direct message propagation between different types of variables for general hybrid Bayesian networks. We

achieve this goal by deriving formulae for exchanging messages under all possible scenarios. Unscented transformation are used to tackle possible nonlinear functional relationships between continuous variables Julier (2002), Sun & Chang (2007a). For arbitrary densities, we proposed to use Gaussian mixture as the approximation before passing messages. The approach does not require any graph transformation, or any numerial integrations. In the framework, each node in the networks propagates messages to its neighbors. Messages are computed locally based on the node types under various circumstances without global knowledge. To maintain scalability, we also propose to use Gaussian mixture reduction techniques Kuo-Chu Chang & Smith (2010), Chang & Sun (2010), to limit the number of Gaussian components, while having the approximation error bounded each time. We term this new scalable and distributed approach Direct Message Passing for Hybrid Bayesian Network (DMP-HBN). Further, for general hybrid models with topology such that a discrete node may have continuous parents, one can always use Shenoy (2006) to convert the model into a network with CLG structure, then apply DMP-HBN for inference. This algorithm is able to provide an exact solution for polytree CLGs, and approximate solution by loopy propagation for general hybrid models.

In the rest of this chapter, we focus on describe the details of DMP-HBN. At the end of this chatper, we will also briefly discuss an up-to-date method to find the most probable explanations (MPE) for hybrid Bayesian networks.

2. Direct message passing

This section describes DMP-HBN algorithm in detail. We first briefly review Pearl's original message passing algorithm. We then extend it for general hybrid models.

2.1 Pearl's message passing algorithm

Recall that in a polytree network, any node X d–separates evidence into $\{\mathbf{e}^+, \mathbf{e}^-\}$, where $\mathbf{e}^+$ and $\mathbf{e}^-$ are evidence from the sub-network "above" X and "below" X respectively. Every node in the network maintains two values called λ and π. The λ value of X is the likelihood, defined as:

$$\lambda(X) = P(\mathbf{e}_X^-|X) \tag{1}$$

The π value of X, defined as:

$$\pi(X) = P(X|\ \mathbf{e}_X^+) \tag{2}$$

is the conditional probability distribution of X given $\mathbf{e}_X^+$. It is easy to see that the belief of a node X given all evidence is just the normalized product of its λ and π values:

$$\begin{aligned} BEL(X) &= P(X|\mathbf{e}) = P(X|\mathbf{e}_X^+, \mathbf{e}_X^-) \\ &= \frac{P(\mathbf{e}_X^-|X, \mathbf{e}_X^+)P(X|\mathbf{e}_X^+)P(\mathbf{e}_X^+)}{P(\mathbf{e}_X^+, \mathbf{e}_X^-)} \\ &= \alpha P(\mathbf{e}_X^-|X)P(X|\mathbf{e}_X^+) \\ &= \alpha\lambda(X)\pi(X) \end{aligned} \tag{3}$$

where α is a normalizing constant. In message passing, every node sends λ messages to each of its parents and π messages to each of its children. Based on its received messages, every

node updates its λ and π values correspondingly. The general message propagation equations of Pearl's algorithm are the following Pearl (1988):

$$\pi(X) = \sum_{\mathbf{T}} P(X|\ \mathbf{T}) \prod_{i=1}^{m} \pi_X(T_i) \tag{4}$$

$$\lambda(X) = \prod_{j=1}^{n} \lambda_{Y_j}(X) \tag{5}$$

$$\pi_{Y_j}(X) = \alpha \left[\prod_{k \neq j} \lambda_{Y_k}(X) \right] \pi(X) \tag{6}$$

$$\lambda_X(T_i) = \sum_{X} \lambda(X) \sum_{T_k:\ k \neq i} P(X|\mathbf{T}) \prod_{k \neq i} \pi_X(T_k) \tag{7}$$

where $T = (T_1, T_2, ..., T_n)$ are the parents of node X; $Y = (Y_1, Y_2, ..., Y_m)$ are children of node X; $\lambda_{Y_j}(X)$ is the λ message node X receives from its child Y_j, $\lambda_X(T_i)$ is the λ message X sends to its parent T_i; $\pi_X(T_i)$ is the π message node X receives from its parent T_i, $\pi_{Y_j}(X)$ is the π message X sends to its child Y_j; and α is a normalizing constant.

Equations (4) to (7) are recursive equations, so we need to initialize messages properly to start the message propagation. Again, Pearl's algorithm is originally designed for discrete polytree networks, so these propagation equations are for computing discrete probabilities. When Pearl's algorithm is applied to a pure discrete polytree network, the messages propagated are exact and so are the beliefs of all nodes after receiving all messages. For pure continuous networks with arbitrary distributions, we proposed a method called Unscented Message Passing Sun & Chang (2007a) using a similar framework with different message representations and a new corresponding computation method. However, with both discrete and continuous variables in the model, passing messages directly between different types of variables requires additional techniques.

2.2 Direct message passing between discrete and continuous variables

We focus our research in this paper to the type of hybrid Bayesian networks that have the same network structure as the CLG, named conditional hybrid model (CHM). In a CHM, a continuous node is not allowed to have any discrete child, while it may have arbitrary distributions and nonlinear relationships between variables. We believe that it is not difficult to extend our algorithm to general hybrid models with arbitrary network structure. Therefore in a CHM, the only case we need to consider when exchanging message between different types of variables is when a continuous node has discrete parents. Without loss of generality, suppose that we have a typical hybrid CPD involving a continuous node X with a discrete parent node D and a continuous parent node U, as shown in 1. Messages sent between these nodes are: (1) π message from D to X, denoted as $\pi_X(D)$; (2) π message from U to X, denoted as $\pi_X(U)$; (3) λ message from X to D, denoted as $\lambda_X(D)$; and (4) λ message from X to U, denoted as $\lambda_X(U)$. In addition, each node needs to maintain its λ and π values.

Let us look at these messages one by one, and derive their corresponding formula based on Pearl's traditional message passing mechanism. First, recall from Equation (6), $\pi_X(D)$ can be

computed by substitution:

$$\pi_X(D) = \alpha \left[\prod_{child \neq X} \lambda_{child}(D) \right] \pi(D) \tag{8}$$

where $\lambda_{child}(D)$ is λ message sent to D from each of its children except X, and $\pi(D)$ is the easily computed message sent from the discrete sub-network "above" D. Note that $\lambda_{child}(D)$ is always in the form of a discrete vector. After normalizing, $\pi_X(D)$ is a discrete probability distribution serving as the mixing prior for a Gaussian mixture.

Similarly, but in a different form, $\pi_X(U)$ can be computed as:

$$\pi_X(U) = \alpha \left[\prod_{child \neq X} \lambda_{child}(U) \right] \pi(U) \tag{9}$$

where $\lambda_{child}(U)$ are λ messages sent to U from its continuous children other than X. These λ messages are continuous messages in the form of Gaussian mixtures. $\pi(U)$ is π value of U, and its computation depends on the type of parent nodes it has. The generalized computation of $\pi(X)$ will be described in the next paragraph. Finally, the resulting $\pi_X(U)$ is a normalized product of Gaussian mixtures, resulting in another Gaussian mixture with a greater number of components.

Now for $\pi(X)$, by applying Equation (4) with integral replacing summation for continuous variable, we have,

$$\begin{aligned} \pi(X) &= \sum_D \int_U P(X|D,U)\pi_X(D)\pi_X(U)dU \\ &= \sum_D \left[\pi_X(D) \int_U P(X|D,U)\pi_X(U)dU \right] \end{aligned} \tag{10}$$

where $\pi_X(D)$ and $\pi_X(U)$ are π messages sent from D and U respectively. For a given $D = d$, $P(X|D = d, U)$ defines a probabilistic functional relationship between X and its continuous parent U. The integral of $P(X|D = d, U)\pi_X(U)$ over U is equivalent to a functional transformation of $\pi_X(U)$, which is a continuous message in the form of a Gaussian mixture. In this functional transformation process, we pass each Gaussian component individually to

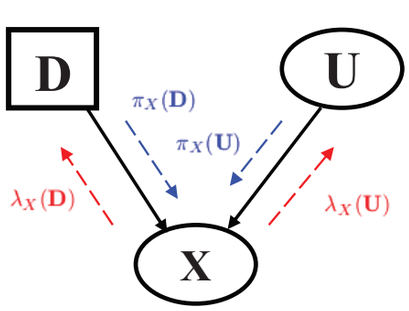


Fig. 1. A typical node with hybrid CPD — continuous node X has discrete parent D and continuous parent U.

form a new Gaussian mixture. Essentially, $\pi(X)$ is a mixture of continuous distributions weighted by $\pi_X(D)$. To avoid the potential for growing complexity of the message, it is possible to approximate the mixture with a single Gaussian density or a Gaussian mixture with fewer components.

$\lambda(X)$ is relatively straightforward to compute as it is the product of λ messages from each of its children, which must be continuous variables due to the network structure restriction. However, since we represent a continuous message as a Gaussian mixture, the product of a set of Gaussian mixtures will be another Gaussian mixture with increased number of components.

Let us now turn to the computation of messages sent from X to its parents D and U. As shown in Equation (7), λ message sent to its parents is essentially an inverse functional transformation of the product of the λ value of the node itself and the π messages sent from all of its other parents via the function defined in the CPD of X. It can be derived as,

$$\lambda_X(D=d) = \int_X \lambda(X) \int_U P(X|D=d,U)\pi_X(U)dUdX \tag{11}$$

where $\int_U P(X|D=d,U)\pi_X(U)dU$ is a functional transformation of a distribution over U into a distribution over X. Further, multiplying by $\lambda(X)$ and integrating over X, results in a non-negative constant, serving as a likelihood of X given $D=d$.

Similarly, the λ message sent from X to its continuous parent U can be expressed as:

$$\begin{aligned}\lambda_X(U) &= \int_X \lambda(X) \sum_D P(X|D,U)\pi_X(D)dX \\ &= \sum_D \left[\pi_X(D) \int_X \lambda(X)P(X|D,U)dX\right]\end{aligned} \tag{12}$$

Note that $\int_X \lambda(X)P(X|D,U)dX$ is an integral of the product of X's λ value and its conditional probability distribution; this integral is over X itself. Therefore it results in a density estimate of its parent multiplied by a coefficient. This coefficient is very critical in computing mixing priors with $\pi_X(D)$ when there is more than one component in the mixture distributions.

Equations (8) to (12) form a baseline for computing messages between discrete and continuous variables. Along with the well-defined formulae for computing messages between the same types of variables, they together provide an unified message passing framework for hybrid Bayesian network models. When the network is a polytree, messages propagated between nodes are exact and so the beliefs. When there are loops in the network, DMP-BN still works in the same way as so-called loopy propagation but provides approximate solution.

To illustrate the algorithm, next we describe in detail the computing process of message passing with a concrete 5-node polytree CLG called $Poly5CLG$. The network structure of $Poly5CLG$ is shown in Figure 2. It consists of 2 discrete node T, C and 3 continuous nodes Y, W, Z. We assume binary discrete nodes and scalar Gaussian continuous nodes in the model. The corresponding CPDs are specified in Figure 3. Suppose leaf nodes C, Z are observable evidence and they are instaniated as state 1, and 5.5 respectively.

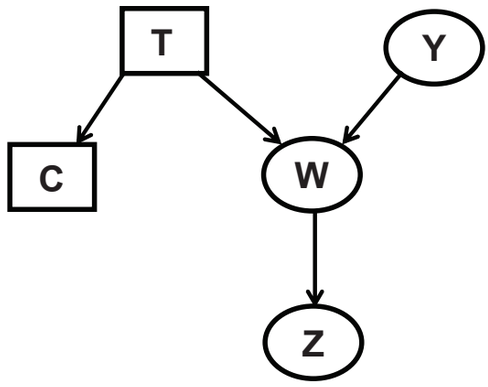


Fig. 2. *Poly*5*CLG*: A demo CLG model consisting of 2 discrete nodes T, C and 3 continuous nodes Y, W, Z.

T: $\boxed{Pr(T=1)=0.5,\ Pr(T=2)=0.5}$

C:

T	$T=1$	$T=2$
$Pr(C=1\|T)$	0.8	0.3
$Pr(C=2\|T)$	0.2	0.7

Y: $\boxed{p(Y)=\mathcal{N}(10,1)}$

W: $\boxed{p(W|T=1)=\mathcal{N}(-1+Y,1)}$ / $\boxed{p(W|T=2)=\mathcal{N}(1+Y,1)}$

Z: $\boxed{p(Z)=\mathcal{N}(0.5W,1)}$

Fig. 3. Nodes CPDs for model *Poly*5*CLG*.

The algorithm is based on iterative computations. First, every node initializes its own λ, π values and messages propagated to its parents (λ messages) and children (π messages). Then, in each iteration, every node updates their λ, π values and messages, utill all of nodes converge to their steady beliefs. For ease of exposition, here we describe the computing process starting from evidence nodes towards their neighbors, and then propagating to other hidden nodes. Further since we only need to know the posterior distributions for hidden variables, we do not compute the messages back to evidence nodes. Starting with the messages from nodes C, Z, it is easily understood that,

$$\lambda(C)=\begin{bmatrix}1\\0\end{bmatrix},\ \lambda(Z)=\begin{cases}\mu=5.5\\ \sigma^2=0\end{cases},$$

where μ, σ^2 are mean and variance representing the continuous message. Then, the λ message sending from C to its parent T can be obtained as

$$\lambda_C(T)=\sum_C \lambda(C)P(C|T)=\sum_C \begin{bmatrix}1&1\\0&0\end{bmatrix}\cdot\begin{bmatrix}0.8&0.3\\0.2&0.7\end{bmatrix}=[0.8\ 0.3],$$

where . is the elementwise multiplication of matrices. Please note that computing λ message sending from Z to its parent W is more complicated and subtle:

$$\lambda_Z(W) = \int_Z \lambda(Z)P(Z|W)dZ.$$

In general, this is essentially an inverse functional transformation for estimating the original dependent variable based on the information of independent variable. Generally, let us assume that the CPD of $P(Z|W)$ is

$$P(Z|W) = \mathcal{N}(f(W), \sigma_0^2),$$

where $f(W)$ is an arbitrary deterministic function specifying the functional relationship between Z and W. Suppose that we now know Z is distributed as $\mathcal{N}(\mu_z, \sigma_z^2)$ (serving as $\lambda(Z)$). Then, $\lambda_Z(W)$ is actually the estimate of W based on this known information about Z, computed as:

$$\begin{aligned}
\lambda_Z(W) &= \int_Z \lambda(Z)P(Z|W)dZ \\
&= \int_Z \mathcal{N}(\mu_z, \sigma_z^2)\mathcal{N}(f(W), \sigma_0^2)dZ \\
&= \int_Z \frac{1}{\sqrt{2\pi}\sigma_z} \exp\left\{-\frac{(Z-\mu_z)^2}{2\sigma_z^2}\right\} \cdot \\
&\qquad \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{-\frac{(Z-f(W))^2}{2\sigma_0^2}\right\} dZ \\
&= \frac{1}{2\pi\sigma_0\sigma_z} \int_Z \exp \\
&\qquad \left\{-\frac{\sigma_0^2(Z-\mu_z)^2 + \sigma_z^2(Z-f(W))^2}{2\sigma_0^2\sigma_z^2}\right\} dZ
\end{aligned} \tag{13}$$

Let us denote the part of exponent in Equation (13) as **E**,

$$\mathbf{E} = \frac{\sigma_0^2(Z-\mu_z)^2 + \sigma_z^2(Z-f(W))^2}{2\sigma_0^2\sigma_z^2}.$$

E can be rearranged to be,

$$\begin{aligned}
\mathbf{E} &= \frac{Z^2 - \frac{2\sigma_0^2\mu_z + 2\sigma_z^2 f(W)}{\sigma_0^2+\sigma_z^2}Z + \frac{\sigma_0^2\mu_z^2 + \sigma_z^2 f^2(W)}{\sigma_0^2+\sigma_z^2}}{\frac{2\sigma_0^2\sigma_z^2}{\sigma_0^2+\sigma_z^2}} \\
&= \frac{(Z-\mathbf{U})^2}{\frac{2\sigma_0^2\sigma_z^2}{\sigma_0^2+\sigma_z^2}} + \frac{(\mu_z - f(W))^2}{2(\sigma_0^2+\sigma_z^2)},
\end{aligned} \tag{14}$$

where $\mathbf{U} = \frac{\sigma_0^2\mu_z + \sigma_z^2 f(W)}{\sigma_0^2+\sigma_z^2}$ is a constant relative to variable Z. Substituting (14) back into (13),

$$\begin{aligned}
\lambda_Z(W) &= \frac{1}{2\pi\sigma_0\sigma_z}\int_Z \exp\left\{-\frac{(Z-\mathbf{U})^2}{\frac{2\sigma_0^2\sigma_z^2}{\sigma_0^2+\sigma_z^2}}\right\} \\
&\quad \exp\left\{\frac{(\mu_z - f(W))^2}{2(\sigma_0^2+\sigma_z^2)}\right\} dZ \\
&= \frac{\sqrt{\sigma_0^2+\sigma_z^2}}{\sqrt{2\pi}\sigma_0\sigma_z}\int_Z \exp\left\{-\frac{(Z-\mathbf{U})^2}{\frac{2\sigma_0^2\sigma_z^2}{\sigma_0^2+\sigma_z^2}}\right\} dZ \\
&\quad \frac{\sqrt{2\pi}\sigma_0\sigma_z}{\sqrt{\sigma_0^2+\sigma_z^2}}\ \frac{1}{2\pi\sigma_0\sigma_z}\exp\left\{-\frac{(f(W)-\mu_z)^2}{2(\sigma_0^2+\sigma_z^2)}\right\} \\
&= \frac{1}{\sqrt{2\pi(\sigma_0^2+\sigma_z^2)}}\exp\left\{-\frac{(f(W)-\mu_z)^2}{2(\sigma_0^2+\sigma_z^2)}\right\}
\end{aligned} \tag{15}$$

$$\begin{aligned}
&= \frac{1}{\sqrt{2\pi(\sigma_0^2+\sigma_z^2)}}\ \sqrt{2\pi}\sigma_w\frac{1}{\sqrt{2\pi}\sigma_w} \\
&\quad \exp\left\{-\frac{(W-\mu_w)^2}{2\sigma_w^2}\right\} \\
&= \frac{\sigma_w}{\sqrt{\sigma_0^2+\sigma_z^2}}\,\mathcal{N}(\mu_w,\sigma_w^2),
\end{aligned} \tag{16}$$

where μ_w, σ_w^2 are the mean and variance estimates for variable W, which always can be obtained by rearranging the exponent in (15) and they must be functions of $\mu_z, \sigma_0, \sigma_z$. In our algorithm, we use unscented transformation to estimate the post distributions for variables undergone nonlinear functions. Note that the constant coefficient $\frac{\sigma_w}{\sqrt{\sigma_0^2+\sigma_z^2}}$ must be part of λ message. It is very critical to keep the coefficient in place while the λ message is in the form of mixture distributions so that it can be updated with correct weights of the components. From (16), it also shows that the λ message is not a distribution. Instead, it is a probabilistic likelihood function. In a special case such that Z is observed as the value z ($\mu_z = z$ and $\sigma_z^2 = 0$), then Equation (15) can be simplified to:

$$\lambda_Z(W) = \frac{1}{\sqrt{2\pi}\sigma_0^2}\exp\left\{-\frac{(f(W)-z)^2}{2\sigma_0^2}\right\} = \frac{\sigma_w}{\sigma_0}\mathcal{N}(\mu_w,\sigma_w^2), \tag{17}$$

where μ_w is a function of z, and σ_w is a function of σ_0. It is straightforward to extend Equation (15), (16), and (17) for continuous node with multiple parents, by adding Gaussian terms from the continuous parents and functions given discrete parents.

Back to the concrete example, substituting actual functions and values into (17),

$$\begin{aligned}\lambda_Z(W) &= \frac{1}{\sqrt{2\pi}\times 1}\exp\left\{-\frac{(0.5W-5.5)^2}{2\times 1}\right\}\\ &= \frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{(W-11)^2}{2\times 4}\right\}\\ &= 2\mathcal{N}(\mu_w=11,\sigma_w^2=4).\end{aligned} \tag{18}$$

Since Z is the only child of W, from Equation (5), we have,

$$\lambda(W)=\lambda_Z(W)=2\mathcal{N}(\mu_w=11,\sigma_w^2=4).$$

For hidden root nodes T, Y, their π values are just their prior distributions,

$$\pi(T)=[0.5\ 0.5],\qquad \pi(Y)=\mathcal{N}(\mu_y=10,\sigma_y^2=1).$$

Now we can compute π messages sending from T, Y to W respectively, according to Equation (8) and (9),

$$\pi_W(T)=\alpha\lambda_C(T)\pi(T)=\alpha[0.8\ 0.3].[0.5\ 0.5]=[0.7273\ 0.2727];$$
$$\pi_W(Y)=\pi(Y)=\mathcal{N}(\mu_y=10,\sigma_y^2=1).$$

Then,

$$\begin{aligned}\pi(W) &= \sum_T\left[\pi_W(T)\int_Y P(W|T,Y)\pi_W(Y)dY\right]\\ &= 0.7273\mathcal{N}(\mu_w=9,\sigma_w^2=2)+0.2727\mathcal{N}(\mu_w=11,\sigma_w^2=2),\end{aligned}$$

which is a Gaussian mixture. So far, W has received all of messages from its parents and children, so,

$$\begin{aligned}BEL(W) &= \alpha\ \lambda(W)\pi(W)\\ &= \alpha\ 2\mathcal{N}(11,4)\ [0.7273\mathcal{N}(9,2)+0.2727\mathcal{N}(11,2)]\\ &= \alpha\ \ [2*0.7273*0.1167\mathcal{N}(9.6667,1.3333)]+\\ &\quad\ \alpha\ \ [2*0.2727*0.1629\mathcal{N}(11,1.3333)]\\ &= 0.6564\mathcal{N}(9.6667,1.3333)+0.3436\mathcal{N}(11,1.3333)\ .\end{aligned}$$

With (11) and (12), the λ messages sending from W to its parents are,

$$\begin{aligned}\lambda_W(T=1) &= \int_W\lambda(W)\int_Y P(W|T=1,Y)\pi_W(Y)dYdW\\ &= \int_W 2\mathcal{N}(11,4)\mathcal{N}(9,2)dW\\ &= 2*0.2334\\ &= 0.4668\ ,\end{aligned}$$

similarly,

$$\begin{aligned}\lambda_W(T=2) &= \int_W \lambda(W) \int_Y P(W|T=2,Y)\pi_W(Y)dYdW \\ &= \int_W 2\mathcal{N}(11,4)\mathcal{N}(11,2)dW \\ &= 2 * 0.3257 \\ &= 0.6514 \ ,\end{aligned}$$

and from (15), (16),

$$\begin{aligned}\lambda_W(Y) &= \sum_T \left[\pi_W(T) \int_W \lambda(W)P(W|T,Y)dX\right] \\ &= 0.7273 \int_W 2\mathcal{N}(11,4)\mathcal{N}(-1+Y,1)dW + \\ &\quad 0.2727 \int_W 2\mathcal{N}(11,4)\mathcal{N}(1+Y,1)dW \\ &= 1.4546\mathcal{N}(12,5) + 0.5454\mathcal{N}(10,5) \ .\end{aligned}$$

Therefore,

$$\begin{aligned}\lambda(T) &= \lambda_C(T)\lambda_W(T) \\ &= [0.8\ 0.3].[0.4668\ 0.6514] \\ &= [0.37344\ 0.19542] \ , \\ \lambda(Y) &= \lambda_W(Y) \\ &= 1.4546\mathcal{N}(12,5) + 0.5454\mathcal{N}(10,5) \ .\end{aligned}$$

Finally the beliefs of nodes T, Y can be computed as,

$$\begin{aligned}BEL(T) &= \alpha\ \lambda(T)\pi(T) \\ &= \alpha\ [0.37344\ 0.19542].[0.5\ 0.5] \\ &= [0.65647\ 0.34353] \ , \\ BEL(Y) &= \alpha\ \lambda(Y)\pi(Y) \\ &= \alpha\ [1.4546\mathcal{N}(12,5) + 0.5454\mathcal{N}(10,5)] \\ &\quad \mathcal{N}(10,1) \\ &= \alpha\ [1.4546 * 0.1167\mathcal{N}(10.3333, 0.8333)] + \\ &\quad \alpha\ [0.5454 * 0.1629\mathcal{N}(10, 0.8333)] \\ &= 0.6564\mathcal{N}(10.3333, 0.8333) + \\ &\quad 0.3436\mathcal{N}(10, 0.8333) \ .\end{aligned}$$

Now, the message passing algorithm provides the posterior distributions for all of hidden nodes T, Y, W. And since this is a poly tree model, the solution is exact.

Note that the presence of discrete parents for continuous variable makes the corresponding continuous messages necessarily a mixture distribution. Unfortunately, the number of

mixture components in the message increases exponentially with the size of joint state space of the discrete parents. In order to scale the algorithm, one alternative is to combine or reduce the mixture components into smaller ones to trade off complexity against accuracy.

2.3 Complexity and scalability

The complexity of exact inference for a hybrid model is essentially determined by the size of the joint state space of all discrete parent nodes (i.e., interface nodes). It is easy to prove that, in a connected CLG, all discrete parents will end up in one clique with at least one continuous node Lerner (2002). Sometimes, even a CLG with very simple structure can give rise to an intractable clique tree. For example, the network shown in Figure 4 will have all of its discrete nodes in one clique, hence making the computations exponential in the size of the joint state space of all discrete nodes. If each discrete node has 10 states, then the resulting clique will have size 10^n, where n is the number of discrete nodes.

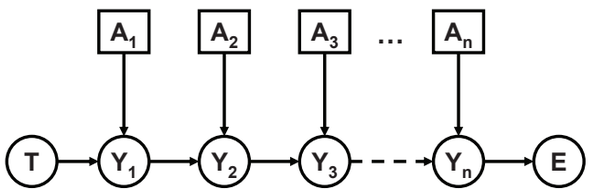


Fig. 4. A simple CLG that has an exponential clique tree: $A_1, A_2, ...A_n$ are discrete nodes, and $T, Y_1, Y_2, ..., Y_n, E$ are continuous nodes.

DMP-HBN has the same problem when exact inference is required. This is because for each state of a discrete parent node, its continuous child has to compute messages according to the function defined in the CPD. Therefore, messages sent by a continuous node with a hybrid CPD will be in the form of a Gaussian mixture in which the components are weighted by probabilities passed from its discrete parents. In particular, as shown in Equaion (10) and (12), $\pi(X)$ and $\lambda_X(U)$ are mixtures of Gaussians with the number of Gaussian components equal to the size of the state space of its discrete parent D. When a mixture message propagates to another continuous node with discrete parents, the message size will increase again exponentially. However, while JT has to deal with this intractability, DMP-HBN has the choice to approximate the original Gaussian mixture with a smaller number of components. In many cases, a Gaussian mixture with significantly fewer components can approximate the original density very well. Let us assume that $f(x)$ is the true density, and $\hat{f}(x)$ is the approximate Gaussian mixture. We use the following distance measure as the metric, called Normalized Integrated Square Error (NISE):

$$d = \frac{\int (f(x) - \hat{f}(x))^2 dx}{\int (f(x))^2 dx + \int (\hat{f}(x))^2 dx} \quad .$$

An example shown in Figure 5 demonstrates a reasonable estimate using only 4 components to approximate a Gaussian mixture with 20 components ($\sqrt{d} < 3\%$). With a pre-defined error bound, Gaussian mixture reduction methods such as the ones proposed in Kuo-Chu Chang & Smith (2010) Schrempf et al. (2005) can be applied to find a good approximate mixture with a smaller number of components. It is straightforward to incorporate these methods into

DMP-HBN to make the algorithm scalable with an acceptable accuracy trade-off. However, it is non-trivial to estimate the overall inference error after the messages are compressed and propagated. In the next section, we will provide some performance results with numerical experiments to evaluate the algorithm under various situations.

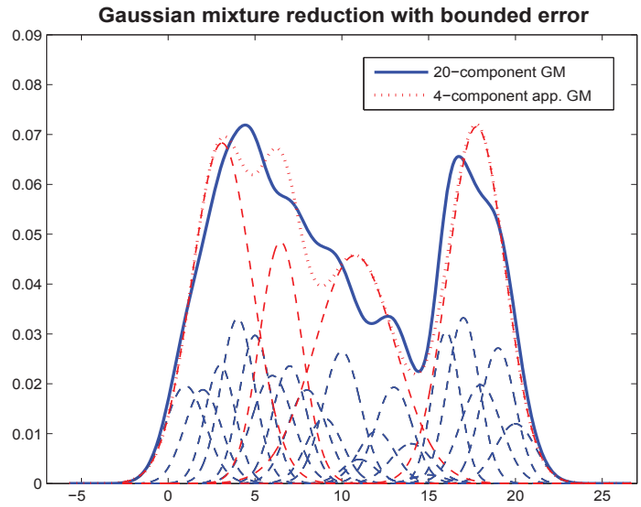


Fig. 5. Using a 4-component GM to approximate a 20-component GM with $\sqrt{d} < 3\%$.

3. Numerical experiments

Theoretically, DMP-HBN can provide exact results for a polytree CLG. For verification purpose, an example model called *Poly12CLG* as shown in Figure 6, was used for the experiment. Assume evidence is observed on leaf nodes E, and Z. With random observations, we conducted more than 30 independent experiments and compared DMP results with the ones obtained by the Junction Tree algorithm. The latter algorithm is considered to be the gold standard and the resulting solutions serve as the ground truth. All experiments show that DMP-HBN provides results identical to the ground truth.

We also conducted scalability tests of DMP algorithm using the same example model *Poly12CLG*. For many decision support applications, the variables of interest tend to be discrete, such as feature identification, entity classifications, or situation hypotheses. In our experiments, we first show how the assessments of hidden discrete nodes in a CLG are affected after collapsing the Gaussian mixture into a single Gaussian when passing messages. We use average absolute probability errors between two discrete distributions as the metric to evaluate the performance. In general, when a node of interest is relatively far away from the evidence, its posterior distribution would not deviate much from its prior. In that case, it is difficult to show the impact of the approximation on the inference error. So we purposely designed CPDs in *Poly12CLG* to move the true posterior probabilites away from its prior. Figure 7 shows the average and maximum errors of the approximate posterior probabilities for hidden discrete nodes V, A, L, B, H, and C, obtained after collapsing Gaussian mixtures into a single term over 100 Monte Carlo simulations. Average and maximum difference between the true posteriors and the priors over these 100 simulations are also shown in the figure for

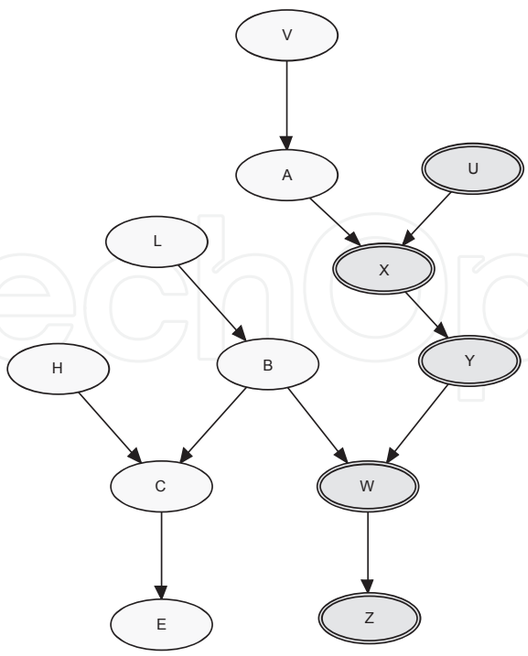
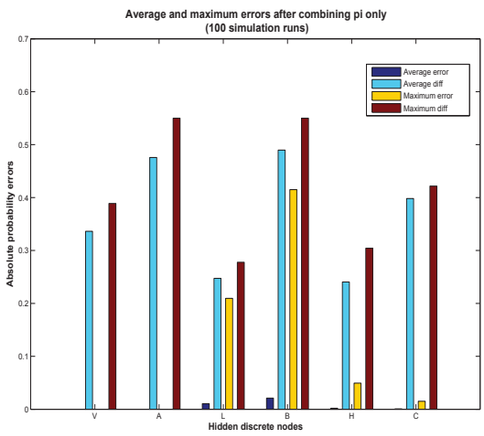


Fig. 6. An example polytree CLG model called Poly12CLG, consisting of 7 discrete nodes V, A, L, B, H, C, E and 5 continuous nodes U, X, Y, W, Z.

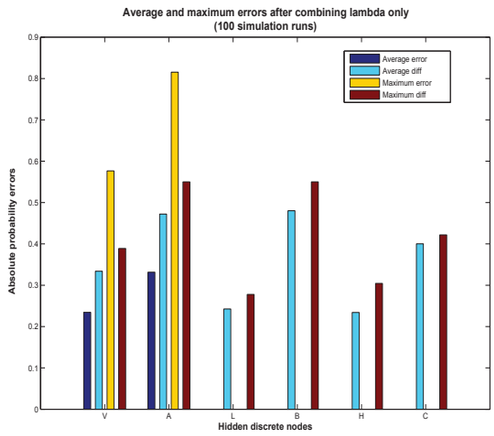
comparison. Figure 7(a) presents the estimate errors when collapsing π values only; Figure 7(b) shows the performance when collapsing λ messages only; and Figure 7 (c) displays the inference errors when collapsing both π values and λ messages whenever a mixture of Gaussians is present.

Notice that reducing the π value of a node does not affect the network "above" it because the π message is being sent downward in the network. Similarly, since a λ message is being sent upward, reducing a λ message will not affect the network "below" the node. For example in Figure 7(a), the posterior probabilities of V and A are exact, and in Figure 7(b), the estimates of L, B, H, and C are also exact without inference error. When reducing both π values and λ messages, all posterior distributions are not exact any more. Results shown in Figure 7(c) suggest that the approximation errors diminish when the nodes are farther away from discrete parents. For example, the approximate errors for nodes L, H, and C are very small. However, discrete parent nodes such as A, and B, are affected significantly. This is not surprising due to the relatively large approximation errors when collapsing a multi-modal Gaussian mixture into a single term. One way to achieve a desired accuracy is to specify a pre-defined error bound whenever we try to reduce a Gaussian mixture into one with fewer components. Although it is difficult to perform theoretical analysis of the total inference error after propagation, it is possible to obtain bounded error if the threshold used is small enough. Figure 8 demonstrates significantly better performance for the same model but with the normalized ISE of the reduced Gaussian mixture limited to less than 5% each time. As can be seen from the figure, the average and maximum errors for all nodes are well less than 1%.

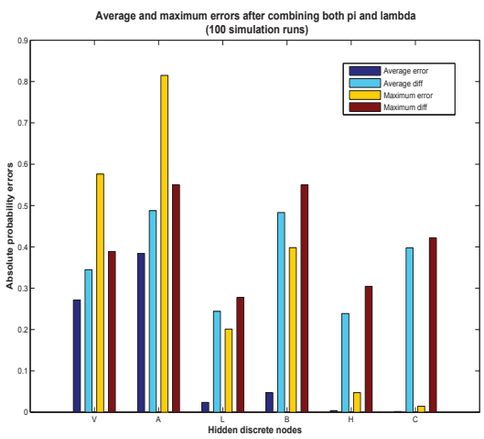
Another example model called *Loop13CLG* (extended from *Poly12CLG*), shown in Figure 9, was used for numerical experimentation on a network with loops. Again, we assume that leaf



(a) combining π values only



(b) combining λ messages only



(c) combining both π values and λ messages

Fig. 7. Scalability test – performance loss after combining Gaussian mixture into one single Gaussian.

nodes E and Z are observable evidence nodes. With random observations, Figure 10 shows the average and maximum absolute errors of posterior probabilities for hidden discrete nodes over 100 Monte Carlo simulations. All simulation runs converge in about 11 iterations. As can be seen from the figure, average approximation errors caused by loopy propagation range from less than 1% to about 5% for hidden discrete nodes.

We also tested DMP with some other networks with randomly pre-defined CPDs. All simulation results suggest that the estimation errors reduce significantly as the node is farther away from the discrete parent nodes.

4. Most probable explanation for hybrid Bayesian networks

In addition to computing the posterior distributions for hidden variables in Bayesian networks, one other important inference task is to find the most probable explanation (MPE).

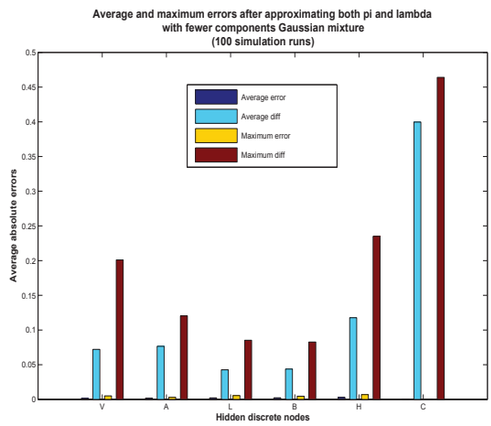


Fig. 8. Accurate estimates of the posterior probabilities resulted by limiting approximation error ($< 5\%$) each time when reducing message with fewer components Gaussian mixture.

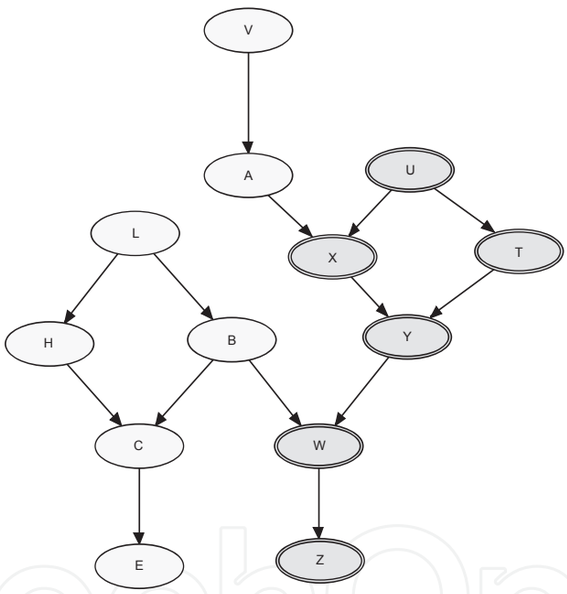


Fig. 9. Loop13CLG – an example CLG model with multiple loops, consisting of 7 discrete nodes V, A, L, B, H, C, E and 6 continuous nodes U, X, TY, W, Z.

MPE provides the most likely configurations to explain away the evidence and helps to manage hypotheses for decision making. In recent years, researchers have proposed a few methods to find the MPE for discrete Bayesian networks. However, finding the MPE for hybrid networks remains challenging. In the following sections, we will briefly describe an up-to-date method to find the MPE in hybrid BNs based on max-product clique tree algorithm.

Let $\mathcal{X}$ represents the full set of variables in a Bayesian network, and $\mathbf{E}$ as a subset of $\mathcal{X}$ containing variables observed, known as evidence. The MPE is the joint assignment of

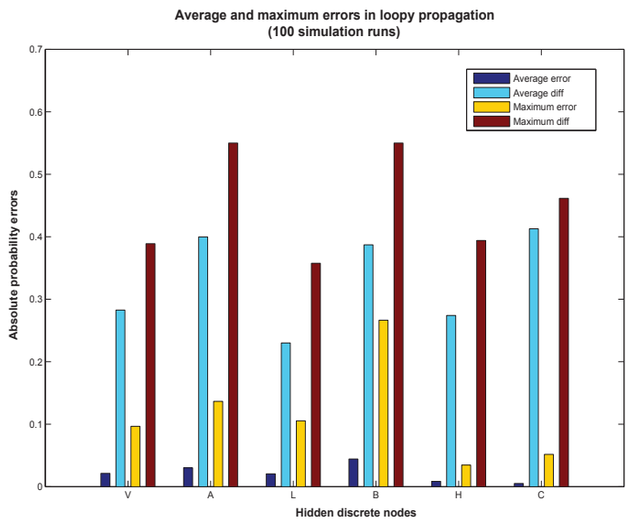


Fig. 10. Performance test with loopy CLG model.

$\mathbf{W} = \mathcal{X} \backslash \mathbf{E}$(subset of all hidden variables) such that:

$$\text{MPE } P(\mathbf{W}|\mathbf{E} = \mathbf{e}) = \arg\max_{\mathbf{w}} P(\mathbf{W} = \mathbf{w}|\mathbf{E} = \mathbf{e}) \tag{19}$$

where $\arg\max_x f(x)$ represents the value of x for which $f(x)$ is maximal.

Note that we have to look at the joint assignment to maximize the joint probability. Individually most likely values of variables that maximize their marginal probabilities are not necessarily part of the MPE. A very simple example is given below to demonstrate this point. Let us look at the BN model consisting of only 3 nodes ($D, E,$ and, F), shown in Figure 11, where $D, E,$ and F are binary discrete random variables with the CPDs listed in the figure.

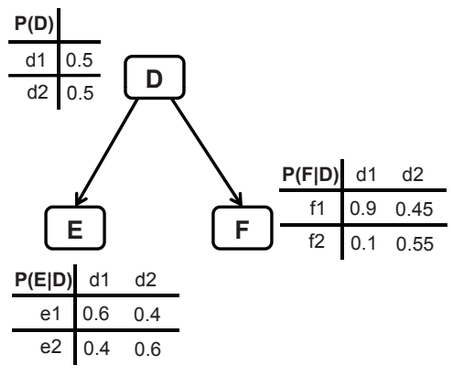


Fig. 11. A simple Bayesian network model consisting of 3 binary discrete nodes ($D, E,$ and, F).

Now let us assume that E is observed as $e2$. It is easy to show that

$$P(D|E = e2) = \begin{bmatrix} d1 : 0.4 \\ d2 : 0.6 \end{bmatrix}, \; P(F|E = e2) = \begin{bmatrix} f1 : 0.63 \\ f2 : 0.37 \end{bmatrix},$$

and

$$P(F|D,E=e2)=P(F|D)=\left[\begin{array}{c|cc} & \text{d1} & \text{d2} \\ \hline \text{f1} & 0.9 & 0.45 \\ \text{f2} & 0.1 & 0.55 \end{array}\right].$$

Therefore

$$P(D,F|E=e2)=P(F|D,E=e2)\times P(D|E=e2)=\left[\begin{array}{c|cc} & \text{d1} & \text{d2} \\ \hline \text{f1} & 0.36 & 0.27 \\ \text{f2} & 0.04 & 0.33 \end{array}\right].$$

From the joint probability distribution, it is clear that the MPE of $E=e2$ is the configuration of $D=d1, F=f1$. If we choose the MPE by individually picking up the values with maximal marginal probabilities, one will end up with a wrong answer as $D=d2, F=f1$, which is obviously not the true MPE.

Theoretically, to compute the maximal joint probability, we have

$$\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{E}=\mathbf{e})=\max_{W_i}\prod_{i=1}^{n}P(W_i|\text{Pa}(W_i),\mathbf{E}=\mathbf{e}) \tag{20}$$

where $W_i(i=1,2,\ldots,n)$ are all of the hidden variables in $\mathbf{W}$ (with the total number of variables in $\mathbf{W}$ being n), and $\text{Pa}(W_i)$ are the parents of node W_i. Clique Tree algorithm has been used in computing the MPE Koller & Friedman (2009), where one needs to replace the marginalization operation with maximization operation for each potential. In this paper, we call this method the *max-product clique tree algorithm*. And accordingly, the potentials in the clique tree are called *max-potentials*.

4.1 Max-calibration of the clique tree for discrete Bayesian networks

The standard clique tree algorithm is a generalization of the variable elimination method for Bayesian network inference. It first transforms the original Bayesian network into a clique tree, which is a undirected poly-tree with cliques serving as nodes in the tree. Each clique is a joint state space of more than one variables, associating with a function called potential. Once a root clique is chosen, one needs to conduct a round trip message propagations in order to have each clique updated by the given evidence. The message propagation from leaf nodes to the root along the path is called upstreaming, also known as collecting evidence; while the opposite is called downstreaming, also known as distributing evidence.

In the process of message propagation between two cliques, a standard protocol is applied. Let us assume that two cliques $\mathbf{C}_i, \mathbf{C}_j$ are neighbors in a clique tree, and separator $\mathbf{S}_{ij}$ is associated with the edge between $\mathbf{C}_i, \mathbf{C}_j$. Potentials for $\mathbf{C}_i, \mathbf{C}_j$ and $\mathbf{S}_{ij}$ are $\phi(\mathbf{C}_i), \phi(\mathbf{C}_j)$ and $\phi(\mathbf{S}_{ij})$ respectively. Sending message from $\mathbf{C}_i$ to $\mathbf{C}_j$ along the separator $\mathbf{S}_{ij}$ follows the message passing protocol presented in Table 1. The sending process is also known as absorption, namely, clique $\mathbf{C}_j$ absorbs information from clique $\mathbf{C}_i$ via their separator $\mathbf{S}_{ij}$.

Note that in Table 1, the first step of message propagation is to marginalizing out the variables in $\mathbf{C}_i$ but not in $\mathbf{C}_j$, so only variables in the separator are left. This is why traditional clique tree algorithm is sometimes called sum-product clique tree method due to this summing out

1. Let $\phi(\mathbf{S}_{ij})' = \sum_{\mathbf{C}_i \setminus \mathbf{S}_{ij}} \phi(\mathbf{C}_i)$, — marginalizing the potential $\phi(\mathbf{C}_i)$ onto the domain of separator $\phi(\mathbf{S}_{ij})$, i.e., projecting it to the domain of separator.
2. Let $\mathcal{L}(\mathbf{S}_{ij}) = \frac{\phi(\mathbf{S}_{ij})'}{\phi(\mathbf{S}_{ij})}$, — dividing the new potential of separator $\phi(\mathbf{S}_{ij})$ by its old one. The ratio $\mathcal{L}(\mathbf{S}_{ij})$ is served as information ratio, also called "likelihood ratio", to update information by filtering out the redundant part.
3. Let $\phi(\mathbf{S}_{ij}) = \phi(\mathbf{S}_{ij})'$, — storing the new potential of the separator for next round message passing.
4. Let $\phi(\mathbf{C}_j) = \phi(\mathbf{C}_j) * \mathcal{L}(\mathbf{S}_{ij})$, — multiplying information ratio from the separator to update potential of $\phi(\mathbf{C}_j)$.

Table 1. Message passing protocol in standard clique tree algorithm

operation. In max-calibration of a clique tree, maximizing replaces marginalizing, while all other operations remain the same in the protocol.

In discrete case, for MPE, it is straightforward to maximize out variables from the joint state distribution. Suppose that we have a joint probability distribution of two binary discrete random variables D, T(states of D, T are $d1, d2, t1$ and $t2$ respectively), shown as below:

$$P(D,T) = \left[\begin{array}{c|cc} & \text{t1} & \text{t2} \\ \hline \text{d1} & 0.32 & 0.16 \\ \text{d2} & 0.39 & 0.13 \end{array}\right].$$

To maximize out T, we have

$$\max_T P(D,T) = \max_T \left[\begin{array}{c|cc} & \text{t1} & \text{t2} \\ \hline \text{d1} & 0.32 & 0.16 \\ \text{d2} & 0.39 & 0.13 \end{array}\right] = \left[\begin{array}{c|c} \text{d1} & 0.32 \\ \text{d2} & 0.39 \end{array}\right]$$

Similarly, if we want to maximize out D from the joint distribution of D, T, it will be:

$$\max_D P(D,T) = \max_D \left[\begin{array}{c|cc} & \text{t1} & \text{t2} \\ \hline \text{d1} & 0.32 & 0.16 \\ \text{d2} & 0.39 & 0.13 \end{array}\right] = \left[\begin{array}{c|c} \text{t1} & 0.39 \\ \text{t2} & 0.16 \end{array}\right]$$

In principle, maximizing out one variable from a joint discrete space returns the marginal maximums in the original joint probabilities along the dimension of this particular variable being maximized over, for all of the configurations of the remaining variables.

With the maximizing substituted in the message propagation protocol, the clique tree will be max-calibrated after conducting the same upstream and downstream message propagations. Then each clique will be updated with the max-potential. After the max-calibration, further maximizations on individual cliques can provide the MPE of all hidden variables. The proofs are very similar to the proofs of the standard clique tree algorithm Jensen (1996) Dawid (1992).

4.2 Finding the MPE for a hybrid Bayesian network using max-product clique tree algorithm

For a hybrid Bayesian network, its clique tree contains at least one hybrid clique, in which both discrete and continuous variables are involved. If we can find a way to conduct maximizing

operations for the hybrid clique, we can then apply the max-calibration process similarly to find the MPE for hybrid BNs.

Let us first take a close look at the hybrid joint space. Without loss of generality, we assume that the continuous variables in the hybrid space are Gaussians. For arbitrary density, theoretically, it is well known that a Gaussian mixture can be used to approximate the original density in any desirable accuracy with sufficient number of components. A simple example of the hybrid space, consisting of one binary discrete variable D with states $d1, d2$, and one scalar Gaussian variable X, is used for demonstration. Assuming that the hybrid joint density is

$$P(D,X) = \left[\begin{array}{c|c} & x \\ \hline d1 & 0.2\mathcal{N}(x;1,0.1) \\ d2 & 0.8\mathcal{N}(x;3,3) \end{array}\right],$$

where $\mathcal{N}(x;u,\sigma^2)$ represents a scalar Gaussian density with mean u, and variance σ^2, and x is a real number. Note that $P(D,X)$ is not a conditional density, nor is a Gaussian mixture, but a hybrid joint density. For example, the joint density for $D = d1, X = 0.5$ is $0.2 * \mathcal{N}(0.5;1,0.1) = 0.325$. If we sum out D, we then can obtain the marginal distribution of X as the linear combination of two Gaussians with weights as $0.2, 0.8$ respectively, which is indeed a Gaussian mixture:

$$P(X) = 0.2\mathcal{N}(x;1,0.1) + 0.8\mathcal{N}(x;3,3).$$

Next, let us see how to maximize one variable from the hybrid joint space using this example. The resulting function after maximizing over some variables is mapped onto the space of the remaining variables. If the variable being maximized out is D from $P(D,X)$, by applying the maximizing rule, we have

$$P(X)^{max} = \max_D P(D,X) = max[0.2\mathcal{N}(x;1,0.1), 0.8\mathcal{N}(x;3,3)],$$

where $P(X)^{max}$ is called the marginal maximum function of X. This is basically a function of x with the values as either $0.2\mathcal{N}(x;1,0.1)$, or $0.8\mathcal{N}(x;3,3)$, whichever is bigger for a given x. As shown in Figure 12, the max of two Gaussians is not a Gaussian mixture. However, because of the closed form of Gaussian density, we can deterministically conclude that the peak of this function is certainly located at one of the mean values among all Gaussian components. Proof is omitted due to its obviousness.

Now let us turn to maximizing out the continuous variable X from the hybrid density $P(D,X)$. Again, by applying the maximizing rule, it is easy to obtain,

$$P(D)^{max} = \max_X P(D,X) = \left[\begin{array}{c|c} d1 & 0.2\mathcal{N}(x=1;1,0.1) \\ d2 & 0.8\mathcal{N}(x=3;3,3) \end{array}\right] = \left[\begin{array}{c|c} d1 & 0.2523 \\ d2 & 0.1843 \end{array}\right],$$

which are the peak points of densities for each weighted Gaussian component given each state of the discrete variable, respectively. In the case of Gaussian density, the peak point is obviously located at the mean, namely, $max[\mathcal{N}(x;u,\sigma^2)] = \mathcal{N}(x=u;u,\sigma^2)$.

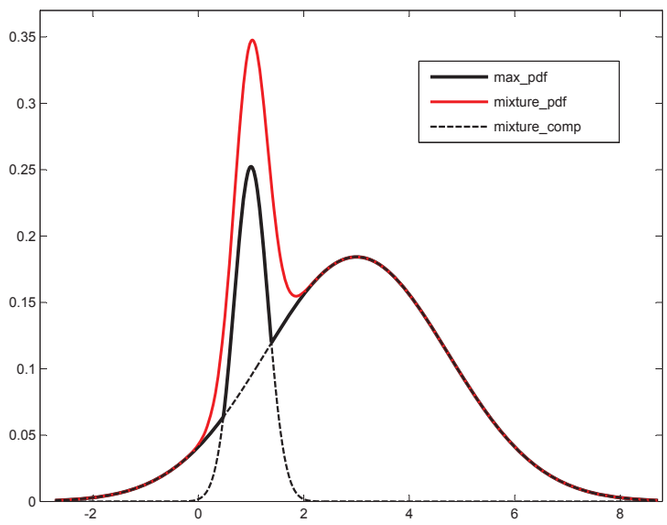


Fig. 12. Maximizing out D from $P(D, X)$ — the resulting function is the max of weighted Gaussian components. In the figure, the red line represents the density of the Gaussian mixture, the bolded black line represents the resulting max function, and the dashed black line shows the original Gaussian components.

Accordingly, the MPE of $P(D, X)$ can be obtained by further examining the value that maximizes the marginal maximum function for each variable. Then,

$$\text{MPE } P(D,X) = \{\arg\max_D P(D)^{max},\ \arg\max_X P(X)^{max}\} = \{D = d1, x = 1\}.$$

At this point, we know how to maximize out variables from both discrete joint space and hybrid space. We still need to know how to maximize out variable from continuous joint space in order to conduct max-calibration for hybrid model. Again, we assume that the continuous variables are Gaussian. Maximizing out a continuous variable from continuous joint space is equivalent to having the value of this variable being its marginal mean and then substituting it into the original joint density function. Let us use a two-dimension Gaussian density to explain the operation. For the sake of simplicity, we assume the two Gaussians X, Y in different dimensions are independent of each other. Namely,

$$P(X,Y) = \mathcal{N}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} u_x \\ u_y \end{bmatrix}, \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}\right),$$

where $\begin{bmatrix} u_x \\ u_y \end{bmatrix}$ is the mean vector, and $\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$ is the covariance matrix. Maximizing out Y from $P(X, Y)$ to obtain the marginal maximum function of X is carried out as below,

$$\begin{aligned} P(X)^{max} &= \max_Y P(X,Y) = \arg\max_Y \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left[\frac{(x-u_x)^2}{\sigma_x^2} + \frac{(y-u_y)^2}{\sigma_y^2}\right]\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma_y}\mathcal{N}(x; u_x, \sigma_x^2). \end{aligned}$$

Given a clique tree of the hybrid model, a strong root of the clique tree, and evidence, this algorithm returns the MPE of the evidence for the original hybrid Bayesian network.

1. Sending messages from leaf cliques to the strong root clique: message passing between cliques follows the protocol shown in Table 1 except using maximizing to replace marginalizing.
2. After the strong root clique receives all messages, sending back messages to all leaf cliques: message passing between cliques follows the protocol shown in Table 1 except using maximizing to replace marginalizing.
3. Conducting further maximizing operation on each clique to obtain the marginal maximum function for each hidden variable, then choosing the value of variable that maximizes its marginal maximum function. Those values together compose the MPE.

Table 2. Hybrid max-product clique tree algorithm to find the MPE for hybrid Bayesian networks

Similar derivation can be done for higher-dimension cases, and/or with dependent variables.

4.3 Division and multiplication between functions

In message passing protocol, shown in Table 1, we also note that division and multiplication operations need to be defined for hybrid models. In this case, the only difference from the discrete case is that how to apply continuous functions in these operations. The result of functional division or multiplication may not have a closed form, but could be computed for any given value of argument variable numerically. And for Gaussian densities, the peak value of the resulting function can be obtained deterministically.

4.4 Hybrid max-product clique tree algorithm (HMP-CT)

Now we are ready to present the hybrid max-product cliquet tree algorithm (HMP-CT) for finding the MPE in hybrid Bayesian networks in Table 2.

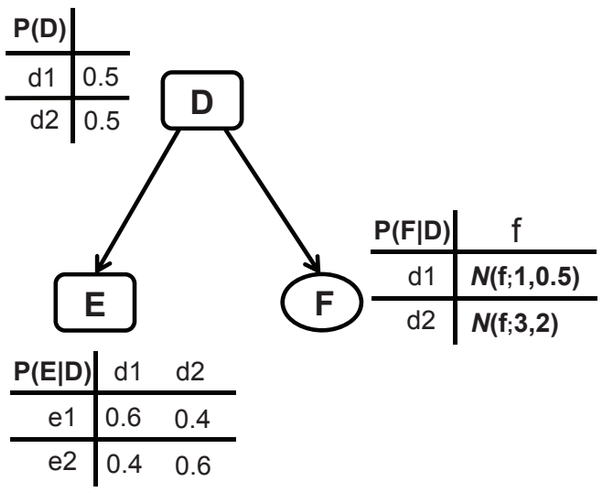
4.5 Numerical example - finding Hybird MPE

In this section, we use a simple hybrid model to demonstrate HMP-CT. With the model shown in Figure 11, we change the node F to be a Gaussian variable and all other parameters and network structure remain the same. The hybrid model with its new CPD is shown in Figure 13(a), where the ellipse is used to represent continuous variable.

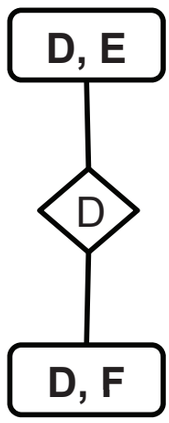
There are only two cliques in the corresponding clique tree of the model, shown in Figure 13(b), in which the clique $\{D, F\}$ is the strong root. Assuming that the observed evidence $E = e2$, let us follow the algorithm described in Table 2 to find the MPE configuration of the hidden nodes D, F.

First, the initial potentials of these two cliques are

$$\phi(D,E) = \left[\begin{array}{c|c} & \text{E=e2} \\ \hline d1 & 0.4 \\ d2 & 0.6 \end{array}\right], \quad \phi(D,F) = \left[\begin{array}{c|c} & f \\ \hline d1 & 0.5\mathcal{N}(f;1,0.5) \\ d2 & 0.5\mathcal{N}(f;3,2) \end{array}\right].$$



(a) network structure and its CPDs



(b) The corresponding clique tree

Fig. 13. A simple hybird Bayesian network model consisting of 2 binary discrete nodes (D, E) and one Gaussian variable (F).

And the potential of the only separator D is uniformly initialized as $\phi(D) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Since the strong root is $\{D, F\}$, the upstreaming message passing is then from the clique $\{D, E\}$ to $\{D, F\}$. We have

$$\phi(D)' = P(D)^{max} = \max_E \phi(D, E) = \left[\begin{array}{c|c} d1 & 0.4 \\ d2 & 0.6 \end{array}\right].$$

The updated potential of D, F is,

$$\phi(D, F) \leftarrow \phi(D, F) \times \frac{\phi(D)'}{\phi(D)} = \left[\begin{array}{c|c} & f \\ \hline d1 & 0.2\mathcal{N}(f; 1, 0.5) \\ d2 & 0.3\mathcal{N}(f; 3, 2) \end{array}\right].$$

Also we need to update the potential of the separator D to be $\phi(D) = \phi(D)' = \left[\begin{array}{c|c} d1 & 0.4 \\ d2 & 0.6 \end{array}\right]$.
Now sending back the message from the root $\{D, F\}$ to the leaf $\{D, E\}$, we have

$$\phi(D)' = \max_F \phi(D, F) = \left[\begin{array}{c|c} d1 & 0.2\mathcal{N}(f = 1; 1, 0.5) \\ d2 & 0.3\mathcal{N}(f = 3; 3, 2) \end{array}\right] = \left[\begin{array}{c|c} d1 & 0.1128 \\ d2 & 0.0846 \end{array}\right].$$

Again, the potential of D, E is updated as,

$$\phi(D, E) \leftarrow \phi(D, E) \times \frac{\phi(D)'}{\phi(D)} = \left[\begin{array}{c|c} & \text{E=e2} \\ \hline d1 & 0.1128 \\ d2 & 0.0846 \end{array}\right].$$

Now the max-calibration of the clique tree is complete. By further maximizing the potentials onto each hidden variable, we have

$$\phi(D)^{max} = \max_{E} \phi(D,E) = \left[\begin{array}{c|c} d1 & 0.1128 \\ d2 & 0.0846 \end{array}\right],$$

and

$$\phi(F)^{max} = \max_{D} \phi(D,F) = max(0.2\mathcal{N}(f;1,0.5), 0.3\mathcal{N}(f;3,2)) = max(0.1596, 0.0598) = 0.1596,$$

located at $f = 1$. Therefore, the MPE of $E = e2$ is $\{D = d1, F = 1\}$. From the joint posterior distribution $P(D,F|E = e2)$, the peak value of joint density associated with the MPE is 0.2257.

5. Summary

In this chapter, we presented a new inference algorithm called DMP-HBN to represent probabilistic messages in the form of Gaussian mixture when continuous variables are involved and allow exchanging messages between discrete and continuous variables directly. This new algorithm provides an alternative for probabilistic inference in hybrid Bayesian networks. It provides full density estimates for continuous variables and can be extended with unscented transformation Sun & Chang (2007a) for the general hybrid models with nonlinear and/or non-Gaussian distributions. Since DMP-HBN is a distributed algorithm utilizing only local information, there is no need to transform the network structure as required by the Junction Tree algorithm. Compared to our previous works in Sun & Chang (2007b), Sun & Chang (2009), that need to partition the hybrid model into different network segments, and then conduct message passing separately, DMP-BN can exchange messages directly between discrete and continuous variables within an unified framework. In addition, the algorithm does not require prior knowledge of the global network topology which could be changing dynamically. This is a major advantage of the algorithm and is particularly important to ensure scalable and reliable message exchanges in a large information network where computations are done locally.

As shown in the empirical simulation results, DMP-HBN is scalable with a performance tradeoff of losing some accuracy. For many decision support applications, we are mainly interested in hidden discrete variables such as entity classifications or high level situation hypotheses. The experimental results show that the estimation errors of the hidden discrete variables depend on the network topology and are relatively modest, especially when the variables of interest are far away from the discrete parent nodes. Theoretically, it is non-trivial to estimate the overall performance bounds quantitatively due to message compressing and propagation. Even though we can have the error bounded each time when we approximate the original Gaussian mixture with less number of components, it is theoretically difficult to estimate the total error after we propagate the approximate messages multiple times. Similar problem exists in filtering for stochastic dynamic systems. This points to an important and very interesting topic for future research.

In addition to the inference task of calculating posterior distributions, finding the MPE is another important type of inference and it has a number of real-life applications in decision support. In the chapter, we introduced and descibed in detail a hybrid max-calibration clique

tree algorithm, called HMP-CT, to find the MPE for hybrid Bayesian networks. We derived all of required operations in the calibration process. Different from the standard sum-product clique tree algorithm, HMP-CT maximizes out variables from the clique potentials instead of marginalizing.

As mentioned in Section 4.3, division and multiplication in message propagation process for hybrid model require functional operations. Further investigations are needed in order to find the better representations of the resulting functions to save computations. In the process, what we need is to obtain the locations (values of variables), where maximize the resulting functions.

To our best knowledge, little research has been done for finding the MPE in hybrid BN models. On the other hand, it is almost inevitable to have continuous variables involved when modeling a real-life problem. It is especially useful to have the MPE for managing multiple most likely hypotheses in many decision support systems. Also, finding the MPE is essentially a global searching problem as to find the maximum. For a genearal optimization problem, if we can decompose the joint state space and model the cost function by a BN-like structure, we can then apply max-calibration algorithm to solve it.

Similarly and with the obvious proof, min-calibration clique tree algorithm can provide the least probable explanation (LPE). In some interesting domains, the LPE is very useful. For example, in prediction market, we always need to know the minimum possible asset of a trader in case that some random events happen to occur that are against this trader's bet.

6. References

Chang, K. C. & Sun, W. (2010). Scalable fusion with mixture distributions in sensor networks, *2010 11th International Conference on Control Automation Robotics & Vision (ICARCV)*, IEEE, pp. 1251–1256.

Charniak, E. (1991). Bayesian networks without tears: making bayesian networks more accessible to the probabilistically unsophisticated, *AI Magazine* 12(4): 50–63.

Cheng, J. & Druzdzel, M. J. (2000). AIS-BN: an adaptive importance sampling algorithm for evidential reasoning in large bayesian networks, *Journal of Artificial Intelligence Research* 13: 155–188.

Cobb, B. R. & Shenoy, P. P. (2006). Inference in hybrid bayesian networks with mixtures of truncated exponentials, *International Journal of Approximate Reasoning* 41: 257–286.

Dawid, A. (1992). Application of a general propagation algorithm for probabilistic expert systems, *Statistics and Computing* 2(1): 25–36.

Fung, R. & Chang, K. C. (1989). Weighting and integrating evidence for stochastic simulation in bayesian networks, *Uncertainty in Artificial Intelligence 5*, Elsevier Science Publishing Company, Inc., New York, pp. 209–219.

Gamerman, D. & Lopes, H. F. (2006). *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*, CRC Press.

Gilks, W. R., Richardson, S. & Spiegelhalter, D. J. (1996). *Markov chain Monte Carlo in practice*, CRC Press.

Jensen, F. (1996). *An Introduction to Bayesian Networks*, Springer-Verlag, New York.

Julier, S. J. (2002). The scaled unscented transformation, *Proceedings of the American Control Conference*, Vol. 6, pp. 4555–4559.

Koller, D. & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, Cambridge, Mass.

Koller, D., Lerner, U. & Angelov, D. (1999). A general algorithm for approximate inference and its application to hybrid bayes nets, *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 324–333.

Kuo-Chu Chang, H. C. & Smith, C. (2010). Constraint optimized weight adaptation for gaussian mixture reduction, *Proceeding of SPIE Conference on Defense, Security, and Sensing*, Orlando,FL.

Lauritzen, S. (1992). Propagation of probabilities, means amd variances in mixed graphical association models, *JASA* 87(420): 1098–1108.

Lerner, U. N. (2002). *Hybrid Bayesian Networks for Reasoning about Complex Systems*, Stanford University.

Murphy, K., Weiss, Y. & Jordan, M. (1999). Loopy belief propagation for approximate inference:an empirical study, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*.

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

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kauffman, San Mateo.

Schrempf, O. C., Feiermann, O. & Hanebeck, U. D. (2005). Optimal mixture approximation of the product of mixtures, *Proceedings of the 8th International Conference on Information Fusion (Fusion 2005)*, Vol. 1, Philadelphia, Pennsylvania, pp. 85–92.

Shachter, R. D. & Peot, M. A. (1999). Simulation approaches to general probabilistic inference on belief networks, *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, Vol. 5.

Shenoy, P. (2006). Inference in hybrid bayesian networks using mixtures of gaussians, *Proceedings of the Twenty-Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, AUAI Press, Arlington, Virginia, pp. 428–436.

Sudderth, E. B., Ihler, A. T., Freeman, W. T. & Willsky, A. S. (2003). Nonparametric belief propagation and facial appearance estimation, *Computer Vision and Pattern Recognition*, Vol. 11, p. 1.

Sun, W. & Chang, K. (2007a). Unscented message passing for arbitrary continuous bayesian networks, *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, Vancouver, Canada.

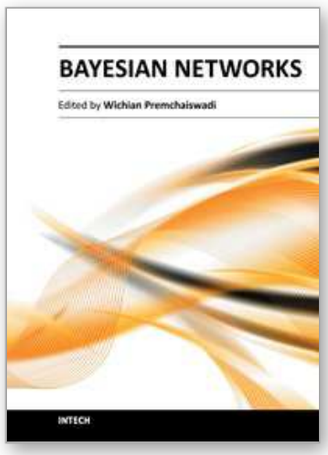
Sun, W. & Chang, K. (2009). Message passing for hybrid bayesian networks: Representation, propagation, and integration, *IEEE Transaction on Aerospace and Electronic Systems* 45: 1525–1537.

Sun, W. & Chang, K. C. (2007b). Hybrid message passing for general mixed bayesian networks, *Proceedings of the 10th International Conference on Information Fusion*, Quebec, Canada.

Wainwright, M. J. & Jordan, M. I. (2008). *Graphical Models, Exponential Families, and Variational Inference*, Now Publishers Inc.

Yuan, C. & Druzdzel, M. J. (2006). Hybrid loopy belief propagation, *Proceedings of the third European Workshop on Probabilistic Graphical Models*, pp. 317–324.

Yuan, C. & Druzdzel, M. J. (2007). Generalized evidence pre-propagated importance sampling for hybrid bayesian networks, *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, AAAI Press, p. 1296–1302. ACM ID: 1619853.



Bayesian Networks

Edited by Dr. Wichian Premchaiswadi

ISBN 978-953-51-0556-5
Hard cover, 114 pages
Publisher InTech
Published online 20, April, 2012
Published in print edition April, 2012

Bayesian Belief Networks are a powerful tool for combining different knowledge sources with various degrees of uncertainty in a mathematically sound and computationally efficient way. A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. When used in conjunction with statistical techniques, the graphical model has several advantages for data modeling. First, because the model encodes dependencies among all variables, it readily handles situations where some data entries are missing. Second, a Bayesian network can be used to learn causal relationships, and hence can be used to gain an understanding about a problem domain and to predict the consequences of intervention. Third, because the model has both causal and probabilistic semantics, it is an ideal representation for combining prior knowledge (which often comes in a causal form) and data. Fourth, Bayesian statistical methods in conjunction with Bayesian networks offer an efficient and principled approach to avoid the over fitting of data.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen