CSCI-6971 Lecture Notes: Graphical models and Belief Propagation*

Kristopher R. Beevers
Department of Computer Science
Rensselaer Polytechnic Institute
beevek@cs.rpi.edu

April 26, 2006

1 Graphical models

We can model the dependencies of random variables using a graph. In a *graphical model*, nodes represent random variables and arcs represent conditional dependencies. (The lack of an arc between two nodes represents a conditional independence assumption.)

We will focus on two types of graphical models: *undirected* models and *directed* models.

- Undirected models are commonly referred to as *Markov random fields* (MRFs). In an MRF, two nodes A and B are conditionally independent given a third node, C, if all paths between A and B go through (are *separated* by) C. Arcs in the graph are, as one might expect, undirected. We introduce MRFs in more detail below.
- Directed models are usually referred to as *Bayesian networks* ("Bayes nets" or BNs for short), or alternatively *belief networks*. Arcs in the graph are directed; one can think of an arc as indicating "causality," i.e., an arc from A to B indicates A "causes" B (but not that B causes A). Associated with each arc is a conditional PDF, e.g., $p(B|A)$.

Graphical models are convenient representations for many applications. We are particularly interested in *inference* on graphical models, i.e., computing marginal PDFs of certain nodes.

1.1 Example: Bayes net

Figure 1 shows an example graphical model for a particular medical diagnosis problem[1] The model is an interpretation of the following rules:

*The primary sources for most of this material are: "Understanding Belief Propagation and its generalizations," J.S. Yedidia, W.T. Freeman, and Y. Weiss, TR-2001-22, Mutsubishi Electric Research Laboratories, January 2002; "A brief introduction to graphical models and Bayesian networks," K. Murphy, 1998, `http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html`; and "Nonparametric Belief Propagation," E.B. Sudderth, A.T. Ihler, W.T. Freeman, and A.S. Willsky, TR P-2551, MIT Lab for Information and Decision Systems, October, 2002.

[1]Borrowed from Yedidia et al., who in turn borrowed it from S.L. Lauritzen and D.J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *J. Royal Stat. Soc. B*, 50, 157–224, 1988.

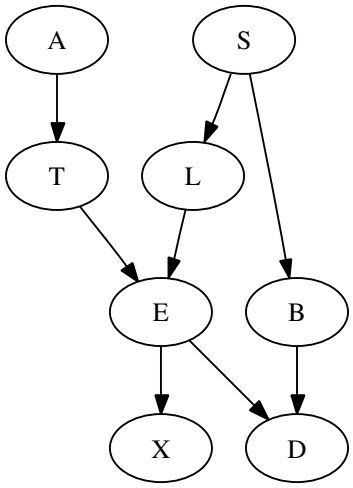

Figure 1: The "Asia" Bayes net (Lauritzen and Spiegelhalter, 1988).

1. A recent trip to Asia Ⓐ increases the chance of contracting tuberculosis Ⓣ.
2. Smoking Ⓢ is a risk factor for both lung cancer Ⓛ and bronchitis Ⓑ.
3. The presence of *either* Ⓔ tuberculosis or lung cancer is detectable by an X-ray Ⓧ, but the X-ray cannot distinguish between them.
4. Dyspnoea Ⓓ (shortness of breath) may be caused by either Ⓔ tuberculosis or lung cancer, or also by bronchitis Ⓑ.

In the graphical representation, each node i represents a random variable or *hidden node* x_i with, in this case, a discrete number of possible states. Associated with each arc in the model is a conditional probability density (in this case a PMF), e.g., $p(x_L|x_S)$, the probability of obtaining lung cancer given that the subject smokes. In the terminology of Bayes nets, S is the *parent* of L; a node may have multiple parents, e.g., D, which we can condition on E and B, i.e., $p(x_D|x_E, x_B)$. Nodes at the "edge" of the graph have no parents, e.g., A or S.

In general, a Bayes net is most useful if it is *sparse* — most nodes have no arcs between them, i.e., they are not statistically dependent.

The joint PDF of all the variables in a Bayes net can be computed as a product of the marginals. In the example:

$$p(\mathbf{x}) = p(x_A, x_S, x_T, x_L, x_B, x_E, x_X, x_D) = \\ p(x_A)p(x_S)p(x_T|x_A)p(x_L|x_S)p(x_B|x_S)p(x_E|x_L, x_T)p(x_D|x_B, x_E)p(x_X|x_E) \quad (1)$$

Definition 1.1. Generalizing, we say: a directed acyclic graph of N nodes defines a Bayes net with N random variables x_i, $i = 1 \ldots N$, that encodes the joint PDF:

$$p(\mathbf{x}) = \prod_{i=1}^{N} p(x_i|\text{par}(x_i)) \quad (2)$$

where $\text{par}(x_i)$ denotes the parents of x_i.

Inference in a Bayes net, which is equivalent to marginalization, can be done (in the discrete case) with:

$$p(x_N) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_{N-1}} p(x_N|x_1, x_2, \ldots, x_{N-1}) \quad (3)$$

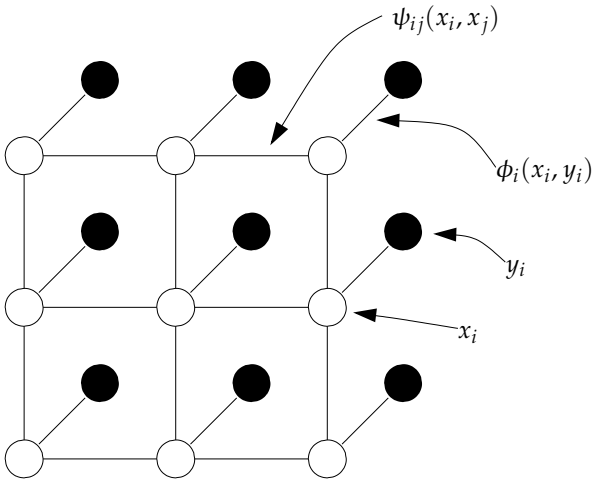

Figure 2: Square lattice pairwise MRF.

Frequently the notation $b(x_N) = p(x_N)$ is used instead to denote the *belief* about x_N. Note that the number of terms in the summation grows exponentially in the number of hidden nodes, so doing inference naïvely is intractable.

1.2 Example: pairwise Markov random field

Consider a computer vision application in which we are given a 1000×1000 pixel grayscale image. We wish to infer quantities x_i for each input (pixel) y_i, e.g., distance to the object in the scene, high-resolution details missing from the image, etc. More generally, we *observe* y_i and wish to infer x_i.

We assume there is some statistical depenency between x_i and y_i. The *evidence* for x_i given by y_i is $\phi_i(x_i, y_i)$.

We also assume there is some underlying structure in the x_i's, e.g., x_i should be "compatible" with nearby x_j's. This is expressed by a *compatibility function* $\psi_{ij}(x_i, x_j)$, which, in a computer vision application, say, only connects nearby pixels.

Figure 2 depicts a square lattice pairwise MRF like that which might arise in a computer vision application.

The MRF model is undirected, which is why we use undirected compatibility functions $\psi_{ij}(x_i, x_j)$ and evidence functions $\phi_i(x_i, y_i)$ instead of "directed" conditional PDFs $p(x_i|x_j)$ and $p(x_i|y_i)$, respectively.

The joint PDF of the inputs y_i and the quantities x_i (e.g., an image and the underlying scene) can be expressed as:

$$p(\mathbf{x}, \mathbf{y}) = \eta \prod_{(ij)} \psi_{ij}(x_i, x_j) \prod_i \phi_i(x_i, y_i) \tag{4}$$

where (ij) is over neighbors in the graphical model, e.g., over nearest neighbors on the square lattice for a computer vision application.

Inference in a pairwise MRF thus involves the computation of $b(x_i)$ for all i, so direct computation takes exponential time in the number of nodes (just as with BNs) — and in applications like computer vision, there are typically very many nodes. We thus need an approximate solution.

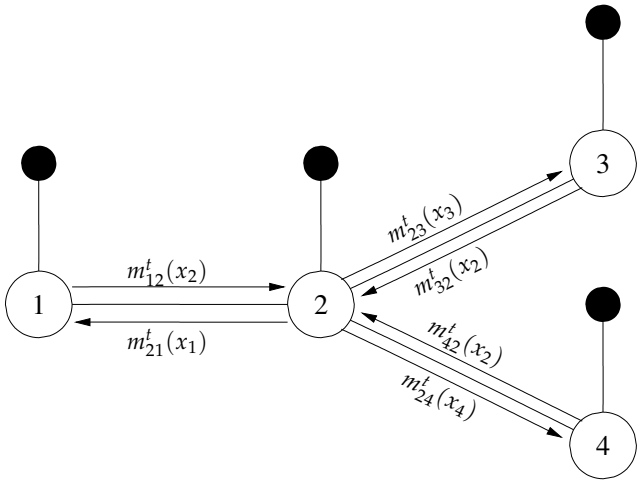

Figure 3: Message propagation in BP.

2 Belief propagation

Belief propagation (BP) is a method for inference on graphical models.[2] We consider the observed (unhidden) variables to be fixed, so we write $\phi_i(x_i)$ as shorthand for $\phi_i(x_i, y_i)$. Thus, our goal is to compute:

$$p(\mathbf{x}) = \eta \prod_{(ij)} \psi_{ij}(x_i, x_j) \prod_i \phi_i(x_i) \tag{5}$$

Definition 2.1. The main idea of belief propagation is to introduce *messages* between hidden nodes in the model. The message $m_{ij}(x_j)$ is a message from hidden node i to hidden node j about what state i "thinks" j should be in. A message is a likelihood function over the state space of x_j:

$$m_{ij}(x_j) \propto p(y_i = \overline{y}_i | x_j) \tag{6}$$

where $\overline{y}_i$ is the predicted input given x_j. Figure 3 depicts the messages in a simple undirected network.

Belief propagation is an iterative algorithm. At each time t, each node sends messages to its neighbors. (We'll add a time index to messages: $m_{ij}^t(x_j)$.) Incoming messages are used to compute the belief at the node. The process is iterated until convergence. The full joint posterior can then be obtained from the product of marginals, or individual marginals can be used in an inference problem.

Belief The belief at node i is proportional to the product of *local evidence* $\phi_i(x_i)$ and all the incoming messages:

$$b_i^t(x_i) = \eta \phi_i(x_i) \prod_{j \in N(i)} m_{ji}^t(x_i) \tag{7}$$

where $N(i)$ denotes the neighbors of i in the graphical model.

[2] We'll focus on pairwise MRFs; it is easy to convert BNs and other graphical models into MRFs (and vice versa) — see Yedidia et al.

Message update rule Messages for the next iteration are computed as:

$$m_{ij}^{t+1}(x_j) \leftarrow \eta \int_{x_i} \phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}^t(x_i)\, dx_i \tag{8}$$

If the MRF is *acyclic* (there are no loops), these two rules are *exact*. We will not prove this but will give an example to show the intuition. Consider the MRF in Figure 3. Here, $b_1(x_t)$ as computed by one iteration of BP is exactly equal to the marginal PDF of x_t:

$$\begin{aligned}
b_1(x_t) &= \eta\phi_1(x_1)m_{21}(x_1) && (9)\\
&= \eta\phi_1(x_1) \int_{x_2} \psi_{12}(x_1, x_2)\phi_2(x_2)m_{32}(x_2)m_{42}(x_2)\, dx_2 && (10)\\
&= \eta\phi_1(x_1) \int_{x_2} \psi_{12}(x_1, x_2)\phi_2(x_2) \\
&\quad \int_{x_3} \psi_{23}(x_2, x_3)\phi_3(x_3) \int_{x_4} \psi_{24}(x_2, x_4)\phi_4(x_4)\, dx_4\, dx_3\, dx_2 && (11)\\
&= \eta \int_{x_2} \int_{x_3} \int_{x_4} p(\mathbf{x}) && (12)\\
&= p(x_1) && (13)
\end{aligned}$$

where Equation 12 follows from the definition of the joint PDF in Equation 4.

An implementation of BP would start with nodes at the edge of the graph and propagate messages, waiting at each node until all messages from incoming edges are available. Exact inference on an acyclic graph thus takes time proportional to the number of edges in the graph, far less than the exponential time required by a naïve computation. The reason is that BP breaks down the "global" computation of marginals into small "local" computations.

2.1 Loopy belief propagation

In practice many graphical models have cycles, as in the graph of Figure 2. *Loopy* belief propagation can be performed on such graphs — with some slight modification. Note that the basic BP algorithm will not work because nodes wait to receive messages from all parents before sending theirs. Instead of waiting, suppose we instead pick some initial messages (at random) and compute local message updates using those. The hope is that eventually the message updates converge and the beliefs approximate the real marginals. As noted by Pearl:[3]

> If we ignore the existence of loops and permit the nodes to continue communicating with each other as if the network were singly connected, messages may circulate indefinitely around these loops, and the process may not converge to a stable equilibrium.

Specifically, statistical dependencies between BP messages are not accounted for in loopy BP, so $\lim_{t\to\infty} b_i^t(x_i) \neq p(x_i)$. In practice, however, loopy BP often works very well even in loopy graphs. Yedida et al. provide some theoretical insight into why — see their paper for details.

[3] J. Pearl, "Probabilistic reasoning in intelligent systems: networks of plausible inference," Morgan Kaufmann, 1988.

2.2 Nonparametric belief propagation

Note that computing the message update (8) is potentially expensive (because of the integration). Often, linear Gaussian models are assumed, and the integral can be easily computed. An alternative is to represent the messages nonparametrically, e.g., with samples, or as a mixture of Gaussians:

$$m_{ij}(x_j) = \sum_{k=1}^{M} w_j^k \mathcal{N}(x_j;\, x_{j'}^i, \Lambda_j) \tag{14}$$

where w_j^k is the weight of the kth Gaussian kernel, x_j^k is the kernel mean, and Λ_j is the "bandwidth" or smoothing parameter, common to all the Gaussians.[4]

The message update (8) can be decomposed into two stages:

1. The computation of the *message products*: $\phi_i(x_i) \prod_{k \in N(i) \setminus j} m_{ki}^t(x_i)$
2. The *propagation* of messages, combining the result of computing the message products with the compatibility potential $\psi_{ij}(x_i, x_j)$ and integrating to produce likelihoods for x_j.

The idea of nonparametric BP is to stochastically approximate these two stages.

2.2.1 Message products

Suppose $\phi_i(x_i)$ and $m_{ki}^t(x_i)$ are all represented by mixtures of weighted Gaussians. The product of d Gaussians is Gaussian, so the product of d Gaussian mixtures each with M components is a Gaussian mixture with M^d components. Every product mixture component is associated with d "labels" $\{l_i\}_{i=1}^d$, where l_i identifies a kernel in the ith mixture. The weight $\overline{w}$ of a product mixture component $\mathcal{N}(x;\, \overline{\mu}, \overline{\Lambda})$ is then:

$$\overline{w} \propto \frac{\prod_{i=1}^d w_i \mathcal{N}(x;\, \mu_i, \Lambda_i)}{\mathcal{N}(x;\, \overline{\mu}, \overline{\Lambda})} \tag{15}$$

where:

$$N(x;\, \overline{\mu}, \overline{\Lambda}) \propto \prod_{\{\bar{l}_i\}_{j=1}^d} \mathcal{N}(x;\, \mu_j, \Lambda_j) \tag{16}$$

with the product being over the labels associated with the product mixture component.

Nonparametric BP samples M times from this product of mixtures to obtain a new M-component mixture, with component means as the samples, and with some fixed smoothing parameter. Explicit sampling from the joint distribution over all d labels requires calculating the M^d component weights and is intractable. However, the conditional distribution of the l_ith label given $\{l_j\}_{j \neq i}$ is simpler (a single mixture of M Gaussians) and we can sample from it in $O(M)$ operations.

Since we can sample from the conditional distributions, we can use Gibbs sampling (see the earlier lecture on MCMC) to draw from the full product mixture. Assuming burn-in time is independent of d and M, we can draw M samples from the product mixture in $O(dM^2)$ operations.

[4] This is the formulation used by Sudderth et al., although it certainly seems reasonable to, say, vary the smoothing parameter per sample. Of course, that complicates the approximation.

2.2.2 Message propagation

To propagate messages, can do Monte Carlo integration. We incorporate $\int_{x_i} \psi_{ij}(x_i, x_j)\, dx_i$ — the *marginal influence* of ψ_{ij} on x_j — into the Gibbs sampler for the message product. Thus, the Gibbs sampler produces samples $x_j^k \sim p(x_j)$.

The samples are propagated to x_i by sampling:

$$x_i^k \sim \eta \psi_{ij}(x_i, x_j^k) \tag{17}$$

for each particle x_j^k produced by the Gibbs sampler. (Equation 17 represents the conditional relationship between x_i and x_j.) The x_i^ks are i.i.d. samples from $m_{ij}^{t+1}(x_j)$ — these are sent, along with a bandwidth to represent a Gaussian mixture, as the message.