Stable Directed Belief Propagation in Gaussian DAGs using the auxiliary variable trick

David Barber
IDIAP Research Institute
Rue du Simplon 4
CH-1920 Martigny, Switzerland
david.barber@idiap.ch

Peter Sollich
Department of Mathematics
King's College,
London WC2R 2LS, U.K
peter.sollich@kcl.ac.uk

Abstract

We consider approximate inference in the important class of large Gaussian distributions corresponding to multiply-connected directed acylic networks. We show how Directed Belief Propagation can be implemented in a numerically stable manner by associating backward (λ) messages with an auxiliary variable, enabling intermediate computations to be carried out in moment form. We apply our method to the Fast Fourier Transform network with missing data, and show that the results are more accurate than those obtained using Undirected Belief Propagation on the equivalent pairwise Markov network.

1 Gaussian Directed Acyclic Graphs (GaussDAGs)

A Gaussian DAG is a distribution over a set of vectors,

$$p(x_1, \ldots, x_N) = \prod_i p(x_i | x_{\mathrm{pa}(i)})$$

in which each local conditional distribution $p(x_i|x_{\mathrm{pa}(i)})$ is a Gaussian. Here $\mathrm{pa}\,(i)$ denotes the variables in the parental set of i. Each conditional Gaussian may be defined by the stochastic linear relation

$$x_i = \sum_{l \in \mathrm{pa}(i)} W_{i \leftarrow l} x_l + \eta_i, \qquad \eta_i \sim \mathcal{N}\left(\mu_i^0, \Sigma_i^0\right) \tag{1}$$

Perhaps the most celebrated instance of a GaussDAG is the Kalman Filter [1, 2]. More recently GaussDAGs have appeared as interesting models in spatio-temporal filtering [3] and the Fast Fourier Transform [4]. Our interest in this paper is a stable version of approximate inference using Belief Propagation in massive multiply-connected networks where exact inference is impractical.

In a typical application, the variables x are split into a set of hidden and visible variables $x = (h, v)$ and inference corresponds to the computation of the conditional $p(h|v)$. Formally, inference is straightforward, even in cases where the graph is multiply connected, since in GaussDAGs the posterior means and covariances are given by

$$\mu_{h|v} = \mu_h + \Sigma_{hv}\Sigma_{vv}^{-1}\left(v - \mu_v\right), \qquad \Sigma_{h|v} = \Sigma_{hh} - \Sigma_{hv}\Sigma_{vv}^{-1}\Sigma_{vh} \tag{2}$$

Here Σ_{hv} represents the h, v block of the joint covariance of $p(h, v)$. Similarly, μ_h and μ_v are the corresponding mean components of $p(h, v)$. These means and covariances are easily found by forward propagation of the recursions equation (1). Hence exact inference is bounded by $O(|h|^3)$ since this is the complexity of matrix inversion. However, in many applications, such as the Kalman Filter, this is unacceptably large since it would mean that computational effort grows cubically with time. One can often exploit the structure to reduce this computation using the Junction Tree algorithm or, equivalently in singly-connected graphs, Belief Propagation (BP).

Our interest in this paper is approximate inference when exact inference using the Junction Tree algorithm is deemed impractical – for example in Kalman Filters with a very large hidden dimension. BP is an attractive approximate algorithm in this case since, if it converges (in either its directed or undirected manifestation) then the inferred means $\mu_{h|v}$ will be exact, although the covariance $\Sigma_{h|v}$ will typically be overconfident [5]. Exactness of the means is an important advantage of BP over other techniques such as approximate block factorisations (see e.g. [6]). Linear algebra solutions based on conjugate gradient methods are also interesting for computing the (in principle exact) posterior means, although they are less easy to apply to computing the posterior covariances.

One approach to using BP is to convert the GaussDAG to an undirected pairwise Markov network, $p(x) \propto \prod_{ij} \psi_{ij}(x_i, x_j) \prod_i \psi_i(x_i)$ for suitable $\psi_{ij}(x_i, x_j) \equiv \exp\left(-x_i^T A_{ij} x_j\right)$ and $\psi_i(x_i) \equiv \exp\left(-\frac{1}{2}\left(x_i^T A_{ii} x_i - 2x_i^T a_i\right)\right)$. For example the directed two-node network $p(x_1, x_2) \propto \exp(-\frac{1}{2}(x_2 - W x_1)^2/\sigma_2^2) \exp(-\frac{1}{2}x_1^2/\sigma_1^2)$ has $a_1 = a_2 = 0$ and matrices $A_{11} = I/\sigma_1^2 + W/\sigma_2^2$, $A_{12} = -W^T/\sigma_2^2$ and $A_{22} = I/\sigma_2^2$; in general one obtains additional links ψ_{ij} between common parents of any node. *Undirected* BP (UBP) can then be run on the resulting network [5] by using the message recursions $\gamma_{i\to j}(x_j) \propto \int_{x_i} \psi_{ij}(x_i, x_j)\psi_i(x_i) \prod_{k\in N(i)\setminus j} \gamma_{k\to i}(x_i)$, where $N(i)$ are the neighbours of node i on the Markov network; the posterior means are given by $p(x_i) \propto \psi_i(x_i) \prod_{k\in N(i)} \gamma_{k\to i}(x_i)$. Our main concern with this approach is that if covariances are small (or even zero), then the conversion will introduce numerical errors. Such issues are important in large networks, such as Kalman Filters, where numerical instability can easily arise from accumulated errors [1].

Directed Belief Propagation

The classical Directed Belief Propagation (DBP) algorithm sends messages from a node to its children and parents (see fig. 1) according to the recursions [7]

$$\rho_{i\to j}(x_i) = \int p(x_i|x_{\mathrm{pa}(i)}) \prod_{l\in\mathrm{pa}(i)} \rho_{l\to i}(x_l) \prod_{k\in\mathrm{ch}(i)\setminus j} \lambda_{k\to i}(x_i)$$

$$\lambda_{i\to l}(x_l) = \int p(x_i|x_{\mathrm{pa}(i)}) \prod_{l'\in\mathrm{pa}(i)\setminus l} \rho_{l'\to i}(x_{l'}) \prod_{k\in\mathrm{ch}(i)} \lambda_{k\to i}(x_i)$$

where the integrals are over all variables except the one on which the message depends. We choose to parameterise the ρ messages using the moment representation

$$\rho_{i\to j}(x_i) \propto \exp -\frac{1}{2}\left(x_i - f_{i\to j}\right)^T F_{i\to j}^{-1}\left(x_i - f_{i\to j}\right)$$

and the λ messages using the canonical representation

$$\lambda_{i\to l}(x_l) \propto \exp -\frac{1}{2}\left(x_l^T G_{i\to l} x_l - 2x_l^T g_{i\to l}\right) \tag{3}$$

Whilst there is a choice about whether to parameterise ρ in either moment or canonical form, no such choice exists for the λ messages since G is generally not invertible. DBP then corresponds to updating the parameters F, f, G, g.

If one carries out the integrals naively, then inverse noise covariances appear explicitly. For example, consider a simple network $k \leftarrow i \leftarrow l$ for which we wish to calculate the message $\lambda_{i\to l}(x_l)$. If $p(x_i|x_l)$ is Gaussian, $\mathcal{N}(Wx_l, \Sigma)$, this is given by

$$\lambda_{i\to l}(x_l) \propto \int dx_i\, p(x_i|x_l) \exp -\frac{1}{2}\left(x_i^T G_{k\to i} x_i - 2x_i^T g_{k\to i}\right)$$

Carrying out this integral naively gives $g_{i\to l} = W^T\Sigma^{-1}\left(G_{k\to i} + \Sigma^{-1}\right)^{-1} g_{k\to i}$ and

$$G_{i\to l} = W^T\left(\Sigma^{-1} - \Sigma^{-1}\left(G_{k\to i} + \Sigma^{-1}\right)^{-1}\Sigma^{-1}\right)W$$

This is numerically problematic in the case of very small (or even zero) noise covariances. Whilst it is possible to rearrange any resulting equations by using Woodbury identities to remove the inverses, there is a simple trick which automatically produces recursions in the correct form, avoiding tedious manipulations in more awkward situations. This trick is potentially of some general benefit, and our main aim is to communicate this to the reader who may find applications also in other areas[1]. The details of how to implement the DBP algorithm are straightforward once the use of the trick has been grasped.

2 The Auxiliary Variable Trick

As we saw above, a naive integration method effectively converts ρ messages to the canonical representation, and performs the integral in this representation. This is not dissimilar to what would happen when converting the directed network to an undirected one. We would prefer to carry out the integrals in a moment representation for ρ. In order to do this we therefore express the λ messages in a form where the canonical variables G, g appear in a moment form, by introducing an *auxiliary variable* $x_{k\to i}$ for each message $\lambda_{k\to i}$. This is defined by

$$x_{k\to i} = G_{k\to i} x_i + \eta_{k\to i} \tag{4}$$

where $\eta_{k\to i} \sim \mathcal{N}(0, G_{k\to i})$. Then equation (3) can be written as

$$\lambda_{i\to l}(x_i) \propto p(x_{i\to l}|x_l)|_{x_{i\to l}=g_{i\to l}}$$

There are different ways to achieve this representation of the λ message as a clamped distribution[2]. In [8] we employed a slightly different form to stabilize Expectation Propagation for switching linear dynamical systems and in [9] we showed how to use this trick for singly-connected Kalman Filters. Here we wish to generalise this to the case of general multiply-connected GaussDAGs.

Normally in DBP, each node sends a λ message to each parent. However, in the case of an evidential node e with only a single parent i and in the absence of noise, $\lambda_{e\to i}(x_i) \propto p(x_e|x_i)$ has an infinite $G_{e\to h}$. To deal with this, we do not parameterise λ messages from evidential children, but rather simply include the relevant factors $p(x_e|x_i)$ directly in the recursions, as we will now describe.

The ρ messages

Using the auxiliary variable method, and separating out evidential variables $e \in E$, we have

$$\rho_{i\to j}(x_i) \propto \int p(x_i|x_{\mathrm{pa}(i)}) \prod_{l\in \mathrm{pa}(i)} \rho_{l\to i}(x_l) \prod_{k\in \mathrm{ch}(i)\setminus j, k\notin E} p(x_{k\to i}|x_i) \prod_{e\in \mathrm{ch}(i)\setminus j, e\in E} p(x_e|x_i)$$

[1]We have recently become aware of a related method in the context of state-space smoothers (M. Briers, personal communication).

[2]E.g. let $x_{i\to l} = Bx_i + \eta_{i\to l}$ be clamped to a, with $\eta_{i\to l} \sim \mathcal{N}(0, A)$. We then require $B^T A^{-1} B = G_{i\to l}$ and $B^T A^{-1} a = g_{i\to l}$. We choose simply $A \equiv B \equiv B^T = G_{i\to l}$.

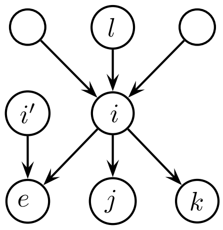

Figure 1: The structure of DBP. The message $\rho_{i\to j}(x_i)$ sends to a child j depends on the parents of i and the parents of any evidential children of i. $\lambda_{i\to l}(x_l)$ depends on the same information and messages as $\rho_{i\to j}(x_i)$, except $\rho_{l\to i}(x_i)$.

where $x_{i\to l}$ is clamped to the value $g_{i\to l}$ and x_e is clamped to value v_e. The usefulness of the auxiliary variable is that we can identify the r.h.s. of the above equation as

$$\rho_{i\to j}(x_i) \propto p(x_i, \{x_{k\to i}\}, \{x_e\})|_{\{x_{k\to i}=g_{k\to i}\},\{x_e=v_e\}}$$

and since the joint distribution on the r.h.s. is Gaussian, the conditional distribution $\rho_{i\to j}(x_i) = p(x_i|\{x_{k\to i} = g_{k\to i}\}, \{x_e = v_e\})$ is easily found using equations (2).

The equations we need to determine the statistics of the joint distribution $p(x_i, \{x_{k\to i}\}, \{x_e\})$ are equation (4) and

$$x_i = \sum_{l\in \mathrm{pa}(i)} W_{i\leftarrow l} x_l + \eta_i, \qquad x_e = W_{e\leftarrow i} x_i + \sum_{i'\in \mathrm{pa}(e)\setminus i} W_{e\leftarrow i'} x_{i'} + \eta_e \tag{5}$$

This gives the means and covariances

$$\mu_i = \sum_{l\in \mathrm{pa}(i)} W_{i\leftarrow l} f_{l\to i} + \mu_i^0, \quad \mu_{k\to i} = G_{k\to i}\mu_i, \quad \mu_e = \mu_e^0 + W_{e\leftarrow i}\mu_i + \sum_{i'\in \mathrm{pa}(e)\setminus i} W_{e\leftarrow i'} f_{i'\to e}$$

$$\begin{aligned}
\Sigma_{ii} &= \sum_{l\in \mathrm{pa}(i)} W_{i\leftarrow l}\Sigma_{l\to i} W_{i\leftarrow l}^T + \Sigma_i^0, \qquad \Sigma_{ki} = G_{k\to i}\Sigma_{ii} \\
\Sigma_{kk} &= G_{k\to i}\Sigma_{ii} G_{k\to i} + G_{k\to i}, \qquad \Sigma_{kk'} = G_{k\to i}\Sigma_{ii} G_{k'\to i}, \\
\Sigma_{ee'} &= W_{e\leftarrow i}\Sigma_{ii} W_{e'\leftarrow i}^T, \qquad \Sigma_{ek} = W_{e\leftarrow i}\Sigma_{ii} G_{k\to i}, \qquad \Sigma_{ei} = W_{e\leftarrow i}\Sigma_{ii} \\
\Sigma_{ee} &= \mu_e^0 + W_{e\leftarrow i}\Sigma_{ii} W_{e\leftarrow i}^T + \sum_{i'\in \mathrm{pa}(e)\setminus i} W_{e\leftarrow l} F_{i'\to e} W_{e\leftarrow i'}^T
\end{aligned} \tag{6}$$

Note that $\Sigma_{ee'}$ contains no correlations between the evidential nodes arising from parents $i' \neq i$ they might share, reflecting the usual DBP prescription that nodes calculate their λ messages separately.

Now that we have the joint distribution $p(x_i, \{x_{k\to i}\}, \{x_e\})$, we need to clamp the $x_{k\to i}$ and x_e into their appropriate states. The structure of this is that we have defined the joint $p(y, z)$ where $y \equiv x_i$ and $z \equiv \{x_{k\to i}\}, \{x_e\}$, and wish to find $p(y|z)$ with z clamped to specific values. In both the mean and covariance, a slight difficulty is that Σ_{zz} may not be invertible; but $\Sigma_{yz}\Sigma_{zz}^{-1}$ remains well defined. In general, the form of Σ_{yz} and Σ_{zz} is (here shown for two non-evidential children k and k' other than j and one evidential child e):

$$\Sigma_{yz} = \begin{pmatrix} \Sigma_{ii} G_{k\to i} & \Sigma_{ii} G_{k'\to i} & \Sigma_{ii} W_{e\leftarrow i}^T \end{pmatrix}$$

We can write the covariance as

$$\Sigma_{zz} = \begin{pmatrix} G_{k\to i}\Sigma_{ii} + I & G_{k\to i}\Sigma_{ii} & G_{k\to i}\Sigma_{ii} W_{e\leftarrow i}^T \\ G_{k'\to i}\Sigma_{ii} & G_{k'\to i}\Sigma_{ii} + I & G_{k'\to i}\Sigma_{ii} W_{e\leftarrow i}^T \\ W_{e\leftarrow i}\Sigma_{ii} & W_{e\leftarrow i}\Sigma_{ii} & \Sigma_{ee} \end{pmatrix} \begin{pmatrix} G_{k\to i} & 0 & 0 \\ 0 & G_{k'\to i} & 0 \\ 0 & 0 & I \end{pmatrix}$$

so that

$$\Sigma_{yz}\Sigma_{zz}^{-1} = \begin{pmatrix} \Sigma_{ii} & \Sigma_{ii} & \Sigma_{ii}W_{e\leftarrow i}^T \end{pmatrix} \begin{pmatrix} G_{k\to i}\Sigma_{ii}+I & G_{k\to i}\Sigma_{ii} & G_{k\to i}\Sigma_{ii}W_{e\leftarrow i}^T \\ G_{k'\to i}\Sigma_{ii} & G_{k'\to i}\Sigma_{ii}+I & G_{k'\to i}\Sigma_{ii}W_{e\leftarrow i}^T \\ W_{e\leftarrow i}\Sigma_{ii} & W_{e\leftarrow i}\Sigma_{ii} & \Sigma_{ee} \end{pmatrix}^{-1}$$

This gives $f_{i\to j} = \mu_y + \Sigma_{yz}\Sigma_{zz}^{-1}(z-\mu_z)$ and covariance $F_{i\to j} = \Sigma_{yy} - \Sigma_{yz}\Sigma_{zz}^{-1}\Sigma_{zy}$ where explicitly $\mu_y = \mu_i$, $\Sigma_{yy} = \Sigma_{ii}$, $z^T = (g_{k\to i}^T, g_{k'\to i}^T, v_e^T)$ and $\mu_z^T = (\mu_k^T, \mu_{k'}^T, \mu_e^T)$.

The λ messages

Using the same auxiliary variable trick as before, the λ messages may be written as

$$\lambda_{i\to l}(x_l) = \int p(x_i|x_{\mathrm{pa}(i)}) \prod_{l'\in\mathrm{pa}(i)\setminus l} \rho_{l'\to i}(x_{l'}) \prod_{k\in\mathrm{ch}(i)\setminus j, k\notin E} p(x_{k\to i}|x_i) \prod_{e\in\mathrm{ch}(i)\setminus j, e\in E} p(x_e|x_i)$$

subject to clamping the auxiliary and evidential variables to their appropriate states. The λ message is then given by the conditional

$$p(\{x_{k\to i}\},\{x_e\}|x_l) \tag{7}$$

This is a little different from the ρ case since we are now interested in the functional dependence on the conditioning variable x_l. We therefore directly isolate the x_l dependent terms in the conditional distribution; from the quadratic form in x_l in the exponent we can then read off the λ messages. The joint distribution of $\{x_{k\to i}\}, \{x_e\}$ (conditional on x_l) is obtained from the relations (4) and (5), except that in x_i we now need to separate off the part dependent on x_l:

$$x_i = \sum_{l'\neq l} W_{i\leftarrow l'}x_{l'} + W_{i\leftarrow l}x_l + \eta_i$$

Explicit expressions for the means and covariances follow directly for this. They involve the mean of x_i, which is $\tilde{\mu} + W_{i\leftarrow l}x_l$ with

$$\tilde{\mu} \equiv \sum_{l'\neq l} W_{i\leftarrow l'} f_{l'\to i} + \mu_i^0$$

as well as the covariance of x_i,

$$\tilde{\Sigma}_{ii} = \sum_{l'\neq l} W_{l'\to i}F_{l'\to i}W_{i\leftarrow l'}^T + \Sigma_i^0$$

For the example case of two non-evidential children k, k' and an evidential child e, the conditional probability (7) is then proportional to

$$\exp -\frac{1}{2}\begin{pmatrix} g_{k\to i} - G_{k\to i}(\tilde{\mu}+W_{i\leftarrow l}x_l) \\ g_{k'\to i} - G_{k'\to i}(\tilde{\mu}+W_{i\leftarrow l}x_l) \\ v_e - \mu_e^0 - W_{e\leftarrow i}(\tilde{\mu}+W_{i\leftarrow l}x_l) \end{pmatrix}^T (MD)^{-1} \begin{pmatrix} g_{k\to i} - G_{k\to i}(\tilde{\mu}+W_{i\leftarrow l}x_l) \\ g_{k'\to i} - G_{k'\to i}(\tilde{\mu}+W_{i\leftarrow l}x_l) \\ v_e - \mu_e^0 - W_{e\leftarrow i}(\tilde{\mu}+W_{i\leftarrow l}x_l) \end{pmatrix}$$

As above for Σ_{zz}, we have decomposed the covariance matrix into

$$M = \begin{pmatrix} G_{k\to i}\tilde{\Sigma}_{ii}+I & G_{k\to i}\tilde{\Sigma}_{ii} & G_{k\to i}\tilde{\Sigma}_{ii}W_{e\leftarrow i}^T \\ G_{k'\to i}\tilde{\Sigma}_{ii} & G_{k'\to i}\tilde{\Sigma}_{ii}+I & G_{k'\to i}\tilde{\Sigma}_{ii}W_{e\leftarrow i}^T \\ W_{e\leftarrow i}\tilde{\Sigma}_{ii} & W_{e\leftarrow i}\tilde{\Sigma}_{ii} & \tilde{\Sigma}_{ee} \end{pmatrix}, \quad D = \begin{pmatrix} G_{k\to i} & 0 & 0 \\ 0 & G_{k'\to i} & 0 \\ 0 & 0 & I \end{pmatrix}$$

where $\tilde{\Sigma}_{ee}$ is obtained by replacing $\Sigma_{ii} \to \tilde{\Sigma}_{ii}$ in (6). To isolate the x_l dependence we define

$$c_k = g_{k\to i} - G_{k\to i}\tilde{\mu}, \quad c_e = e - \mu_e - W_{e\leftarrow i}\tilde{\mu}, \quad U_k = W_{i\leftarrow l}, \quad U_e = W_{e\leftarrow i}W_{i\leftarrow l}$$

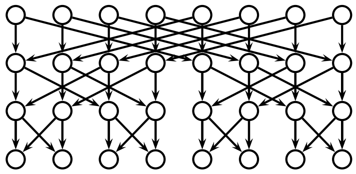

Figure 2: The FFT network for computing the Fourier Transform of an 8 component data vector. The data vector v is clamped in reverse bit order in the bottom layer of the network. The Fourier components (in standard order) are, after inference in this network, given by the bits in the top layer. All nodes below the top layer have zero covariance so that with no missing data the top layer produces the exact FFT. In the case of missing data, the prior in the top layer is used to make the inference well-posed.

Then the above λ message is proportional to $\exp -\frac{1}{2}\left(c - DUx_l\right)^T (MD)^{-1} \left(c - DUx_l\right)$ where $c^T = (c_k^T, c_{k'}^T, c_e^T)$ and $U^T = (U_k^T, U_{k'}^T, U_e^T)$. This gives the message update

$$g_{i\to l} = U^T M^{-1} c, \qquad G_{i\to l} = U^T M^{-1} DU$$

The posterior marginals, finally, are obtained in the standard way by just a slight modification of the ρ message:

$$p(x_i|v) = \int p(x_i|x_{\mathrm{pa}(i)}) \prod_{l\in\mathrm{pa}(i)} \rho_{l\to i}(x_l) \prod_{k\in\mathrm{ch}(i)} \lambda_{k\to i}(x_i)$$

3 Experiments on a Butterfly FFT network

An example of a GaussDAG which contains zero covariances was discussed in [4], and provides an interesting and practical example for comparison of directed versus undirected BP methods. The Fast Fourier Transform FFT(x) normally deals with only the case of complete observations x. An elegant and potentially extremely useful method for dealing with missing data was proposed in [4], in which the one dimensional FFT was considered as a generative GaussDAG with the structure of a so-called butterfly network. This is based on the well-known recursion for computing the FFT of an $n = 2^d$ dimensional vector x in $O(n \log n)$ time. For $W = \exp(-2\pi i/n)$, the k^{th} Fourier coefficient F_k is given by

$$F_k = \sum_{j=0}^{n-1} W^{kj} x_j = \sum_{j=0}^{n/2-1} W^{2kj} x_{2j} + \sum_{j=0}^{n/2-1} W^{(2j+1)k} x_{2j+1} = F_k^e + W^k F_k^o$$

where F_k^e and F_k^o denote the k^{th} component of the length-$n/2$ Fourier transform of the even and components of x_j, respectively. This indicates how the FFT can be recursively computed. The inverse FFT can be computed similarly by replacing W by its complex conjugate. This means that we can generate, from the Fourier coefficients at the top layer, the data points themselves (in reverse bit order) in the bottom layer. The details of exactly how to do this are well explained in [4]. Using a prior on the Fourier coefficients in the top layer, computing the Fourier coefficients in the case of missing data is an inference problem in a GaussDAG in which some of the bottom layer nodes are clamped to their evidential states, and we wish to infer the top layer Fourier coefficients (as well as possibly the missing data in the bottom layer). In our implementation, we represent complex arithmetic using an equivalent two component vector arithmetic. We use a zero mean prior on the Fourier coefficients, and independent isotropic covariances. All noise covariances below the top layer are set to zero. In [4] inference in this network was performed using undirected BP

(UBP) by first transforming the network into a pairwise Markov network. To deal with the problem of zero covariances, a small "jitter" value was substituted instead. The jitter value can have a large effect on the numerical accuracy of UBP, and we used 10^{-11} which was based on experiments with small networks, see fig. 3.

Naive UBP doesn't work well since convergence is hampered by many loops of length four. In our DBP we also found that the tight loops in the network caused difficulties with convergence, and as in [4] used a clustering scheme to ameliorate this. We merge the two children of any parent node into a cluster variable; each node except for those in the top layer is then contained in one such cluster. In the top layer, we cluster nodes with common children, giving again non-overlapping clusters containing two nodes each. In the UBP implementation of [4], nodes that form a loop of length four were merged into four-node clusters (which can overlap each other). The two algorithms, UBP and DBP, then have roughly the same computational complexity per iteration. Our DBP implementation is for general GaussDAGs and so is not fine tuned to take advantage of the layered structure of the FFT network as in [4]. We therefore compare the algorithms in terms of iterations (one update for each message) rather than absolute run time. For a network of $n = 16$ nodes in each layer, we ran 100 experiments with half of the n data points missing at random and the prior FFT variances selected from a uniform distribution on $(0, 1)$. First we ran both UBP and DBP to convergence, defined as the point where the posterior means change by less than 10^{-15} from one iteration to the next. The number of iterations used by both methods to reach this convergence tolerance was roughly the same, differing only by a small number of iterations. However, in about a third of runs UBP struggled to converge to this high tolerance and we then stopped it after a maximum of 50 iterations, which gave similar accuracy to otherwise converged runs. The resulting mean absolute error of the inferred FFT components from the true target values was $(1.5 \pm 1) \times 10^{-8}$ for UBP and $(3.2 \pm 1.4) \times 10^{-14}$ for DBP. Similar experiments give for $n = 32$: $(2.8 \pm 1.1) \times 10^{-8}$ for UBP and $(8.6 \pm 4) \times 10^{-14}$ for DBP. Here UBP usually did not converge within the maximally allowed 100 iterations, whereas DBP always converged within less than 50 iterations. For $n = 64$, both UBP and DBP struggled to converge within 100 iteration, giving errors of $(9.2 \pm 4) \times 10^{-8}$ for UBP and $(2.6 \pm 1.1) \times 10^{-13}$ for DBP. The increase in the errors stresses the importance of numerical accuracy as we increase the network size.

In fig. 3 we also consider the accuracy of the two methods as we increase n at a *fixed* number of iterations, chosen here as 20. The same random prior was used as before, with half the data vector missing at random, and plotted are the mean error and standard deviation over 20 such experiments. As we can see, the performance of both DBP and UBP deteriorates with increasing n, although DBP remains consistently superior by at least two orders of magnitude; this becomes critically important as the network size increases to practically interesting limits. The performance loss for larger networks is partly due to the longer convergence time required for both methods, with 20 iterations not sufficient for convergence. If both methods are run to convergence, then DBP shows a more gradual n-dependence of the mean error which almost tracks the one of UBP but remains at absolute error levels that are lower by 5 to 6 orders of magnitude.

4 Discussion

Directed Gaussian networks are one of the most common forms of continuous graphical models, for which accurate and stable inference techniques are of considerable interest. In cases where exact inference is impractical, BP is a useful approximation since, when it converges, the posterior means are correct. Whilst the two forms of BP – directed and undirected – may be made mathematically equivalent, their numerical stability is different, and may be dramatically so in many practical scenarios where noise covariances are small or even zero. Our approach was to use the generic auxiliary variable trick to easily derive a

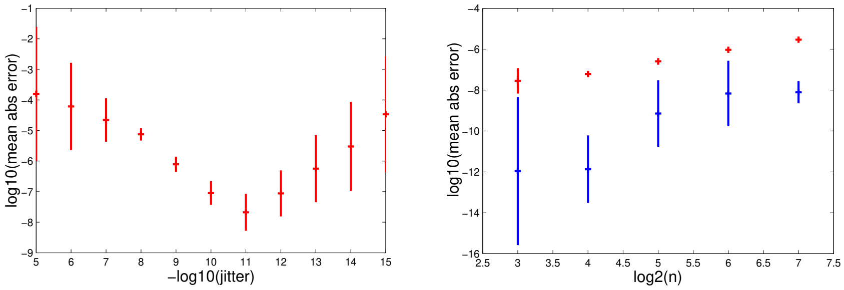

Figure 3: Left: For an FFT network computing the FFT of a 16 dimensional data vector, with half of the components missing at random, the results of the UBP method of [4] are plotted as a function of the jitter. Right: Comparison of the accuracy of the UBP and DBP methods after 20 iterations of each algorithm. Plotted are the mean of the log of the absolute error, together with $\pm$ one standard deviation over the 20 random experiments (see text) for each network size. The upper results are for UBP and the lower ones for DBP.

directed BP implementation without the explicit appearance of inverse noise covariances. The superior performance of directed over undirected Belief Propagation on the FFT network suggests that our directed implementation may be suitable for approximate inference in very large networks such as Kalman Filters with extremely large hidden dimensions and other important practical cases, for which approximate inference may be treated as marginals of a massive multiply-connected graph. MATLAB code for inference in GaussDAGs is available at `http://www.idiap.ch/~barber/GaussDAGs.zip`.

Acknowledgment

We are very grateful to Amos Storkey for making his UBP FFT code available to us, and for helpful discussions.

References

[1] M. Verhaegen and P. Van Dooren. Numerical Aspects of Different Kalman Filter Implementations. *IEEE Transactions of Automatic Control*, 31(10):907–917, 1986.

[2] Y. Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking : Principles, Techniques and Software*. Artech House, Norwood, MA, 1998.

[3] A. Galka, O. Yamashita, T. Ozaki, R. Biscay, and P. Valdes-Sosa. A solution to the dynamical inverse problem of EEG generation using spatiotemporal Kalman filtering. *NeuroImage*, pages 435 – 453, 2004.

[4] A. J. Storkey. Generalised Propagation for Fast Fourier Transforms with Partial or Missing Data. In *Neural Information Processing Systems 16*, 2004.

[5] Y. Weiss and W.T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10), 2001.

[6] M. Verlaan and A. W. Heemink. Reduced Rank Square Root Filters for Large Scale Data Assimilation Problems. *Second International Symposium on Assimilation of Observations in Meteorology and Oceanography, World Meteorological Organization*, pages 247–252, 1995.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, 1988.

[8] D. Barber and B. Mesot. Construction and comparison of approximations for Switching Linear Gaussian State Space Models. Technical Report 05:06, IDIAP Research Institute, February 2005.

[9] D. Barber. The Auxiliary Variable Trick for deriving Kalman Smoothers. Technical Report 04:87, IDIAP Research Institute, December 2004. Submitted to IEEE Trans. Automatic Control.