Maximum-likelihood learning of cumulative distribution functions on graphs

Jim C. Huang and Nebojsa Jojic
Microsoft Research
Redmond, WA 98052
jimhua,jojic@microsoft.com

Abstract

For many applications, a probability model can be more easily expressed as a cumulative distribution functions (CDF) as compared to the use of probability density or mass functions (PDF/PMFs). One advantage of CDF models is the simplicity of representing multivariate heavy-tailed distributions. Examples of fields that can benefit from the use of graphical models for CDFs include climatology and epidemiology, where data follow heavy-tailed distributions and exhibit spatial correlations so that dependencies between model variables must be accounted for. However, in most cases the problem of learning from data consists of optimizing the log-likelihood function with respect to model parameters where we are required to optimize a log-PDF/PMF and not a log-CDF. Given a CDF defined on a graph, we present a message-passing algorithm called the gradient-derivative-product (GDP) algorithm that allows us to learn the model in terms of the log-likelihood function whereby messages correspond to local gradients of the likelihood with respect to model parameters. We demonstrate the GDP algorithm on real-world rainfall and H1N1 mortality data and we show that the heavy-tailed multivariate distributions that arise in these problems can both be naturally parameterized and tractably estimated from data using our algorithm.

1 Introduction

Probabilistic graphical models are widely used for compactly representing joint PDFs/PMFs. While in many applications the PDF/PMF is a natural probabilistic representation, in many other situations the probability model of interest may be difficult to specify in this form. For example, in domains such as climate and infectious disease modeling (Coles, 2001), we may wish to model multivariate heavy-tailed distributions, many of which are more naturally described as *cumulative distribution functions* (CDFs) as compared to parameterizations using PDF/PMFs (Huang, 2009). Such distributions are often highly non-Gaussian (see Figure 1 for an example) and include, but are not limited to, extreme-value distributions, or *max-stable processes* (Coles, 2001). The problem of constructing multivariate heavy-tailed distributions and that of learning from data often leads to computational difficulties. This difficulty is compounded by the need to account for dependencies among several variables in the model, where many of these dependencies may result from the shared influence of latent variables with unknown distributions and structure. Recently, cumulative distribution networks (CDNs) were proposed as a class of graphical models for representing joint CDFs (Huang and Frey, 2008; Huang, 2009). These models were used to construct CDFs defined over many random variables as a product of functions defined over subsets of variables. Inference in CDNs corresponds to computing mixed derivatives of the joint CDF with respect to subsets of variables in the CDN. An efficient message-passing algorithm called derivative-sum-product (DSP) was developed by (Huang and Frey, 2008) for efficiently computing such mixed derivatives. The ability to efficiently perform inference in

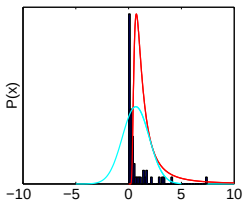


Figure 1: Example of an extreme-value distribution (red) and a Gaussian distribution (cyan) fit to rainfall measurements.

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

CDNs, coupled with the graphical notation afforded by CDNs, make these models well-suited for modeling high-dimensional CDFs.

The problem of estimating the parameters of CDNs from data is, however, less straightforward. A CDN represents a joint CDF: as a result, in order to learn a model from data according to maximum-likelihood (with some exceptions in which the likelihood itself is a CDF: see (Huang and Frey, 2009, see)), we require a likelihood function in the form of a PDF/PMF. In this paper, we take advantage of the graphical modelling framework of CDNs to propose a message-passing algorithm called gradient-derivative-product (GDP) for computing gradients of the model likelihood with respect to model parameters. We derive the algorithm starting from the DSP algorithm, which provides us with mixed derivatives of the joint CDF. We apply the GDP algorithm to the problem of learning models for A) rainfall, where we are given spatial rainfall measurements and B) influenza epidemics where we are given spatial mortality indices. We show that the resulting max-stable processes can be naturally described by CDNs and that the GDP algorithm provides a means to efficiently perform maximum-likelihood estimation of model parameters.

2 Cumulative distribution networks

In this section we briefly review previous work on CDNs and the DSP algorithm for inference and differentiation in CDNs. A CDN is an undirected bipartite graph $\mathcal{G} = (V, S, E)$ consisting of a set of variable nodes $\alpha \in V$, function nodes $s \in S$ defined over neighboring variable nodes $\mathcal{N}(s)$, a set of undirected edges linking variables to functions and a specification for each function ϕ_s in the model. The joint CDF $F(\mathbf{x})$ represented by $\mathcal{G}$ is given by

$$F(\mathbf{x}) = \prod_{s \in S} \phi_s(\mathbf{x}_s), \tag{1}$$

where each of the functions $\phi_s : \mathbb{R}^{|\mathcal{N}(s)|} \mapsto [0, 1]$ satisfies the properties of a CDF. Without loss of generality we will assume that each function in the CDN is connected to two variable nodes α, β so $\phi_s(\mathbf{x}_s) = \phi_s(x_\alpha, x_\beta)$. An example of a CDN is given in Figure 2(a), where diamonds represent CDN functions and circles represent variables. The properties satisfied by the functions ϕ_s include $\partial_{\mathbf{x}_A}\left[\phi_s(\mathbf{x}_s)\right] \geq 0 \; \forall s \in S, A \subseteq \mathcal{N}(s)$, where $\mathbf{x}_A$ is the set of variables corresponding to node set A and $\partial_{\mathbf{x}_A}\left[\cdot\right]$ is the mixed derivative operator with respect to variables $\mathbf{x}_A$. For a CDF, the PDF/PMF is defined in terms of the CDF as $P(\mathbf{x}) = \partial_{\mathbf{x}}\left[F(\mathbf{x})\right]$.

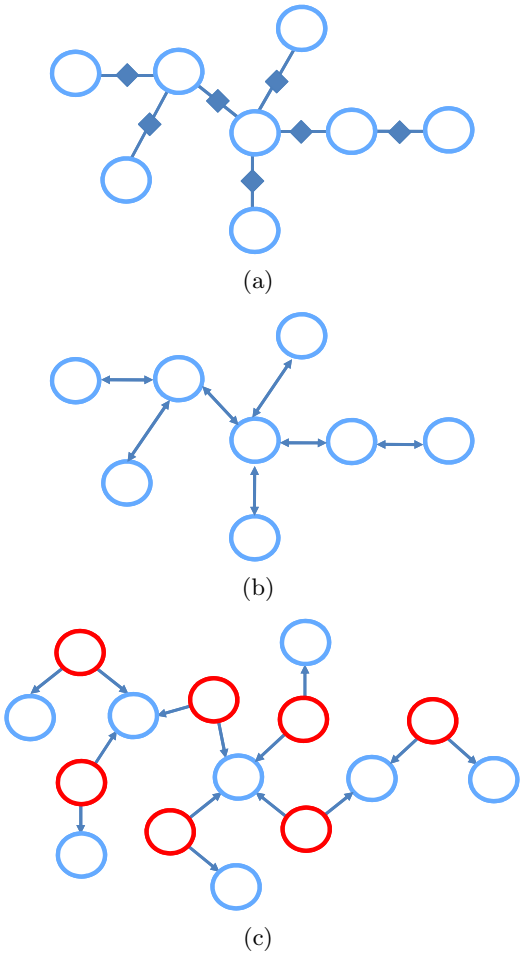


Figure 2: a) A CDN; b) A BDG, in which the absence of a bi-directed edges between two variable nodes indicates marginal independence between the corresponding random variables. In a CDN, variables whose nodes do not have any neighboring functions in common are marginally independent. Both graphs in a) and b) represent the same set of marginal independence constraints amongst variables; c) The canonical DAG associated with the BDG of b). The dependencies among variables in the BDG result from the shared influence of latent variables in the canonical DAG (shown in red).

2.1 Connection to bi-directed graphical models

For CDNs defined over continuous random variables, it can be shown (Huang, 2009) that the resulting conditional independence relationships between subsets of variables in the model are the same as those for bi-directed graphs (BDGs) (Drton and Richardson, 2004; Richardson, 2003). In both cases, the absence of a direct link between two variables indicates *marginal* independence of those two variables in the model. In a CDN, two variables that share a function node in the graph are considered to be directly linked, so that variables that share no functions nodes in common are marginally independent. An example of a BDG and a CDN that model the same set of marginal indepen-

dence constraints is shown in Figures 2(a), 2(b). BDGs can be also viewed as modeling the set of marginal independence constraints among variables that would be obtained from a canonical directed acyclic graphical (DAG) model in which dependencies between variables in the model result from the shared influence of latent variables (Figure 2(c)). Both CDNs and BDGs are well-suited to settings in which dependencies in the model can be explained by the influence of additional latent variables, but no detailed knowledge about these latent variables is available (Drton and Richardson, 2004). For example, in modeling rainfall, latent variables might correspond to shared weather systems that span multiple sites.

2.2 The derivative-sum-product algorithm

In a CDN, the problem of inference, or computing conditional CDFs, is equivalent to computing higher-order derivatives of the joint CDF with respect to subsets of variables (Huang, 2009). Computing derivatives of the form $\partial_{\mathbf{x}_A}\left[F(\mathbf{x})\right]$ for some subset of nodes $A \subseteq V$ can be solved efficiently by the derivative-sum-product (DSP) algorithm in which messages correspond to mixed derivatives with respect to subsets of variables in the CDN. The intuition behind the algorithm is that we reduce a global differentiation operation to a series of local derivative computations, each of which takes the form of a message. For tree-structured CDNs, the messages computed by the DSP algorithm are guaranteed to correspond to the correct mixed derivatives (Huang, 2009). In the case of CDNs containing functions of two arguments, the updates for the DSP algorithm are given for each variable node $\alpha \in V$ and neighboring function node $s \in S$ as

$$\mu_{\alpha\to s}(\mathbf{x}) = \prod_{s'\in\mathcal{N}(\alpha)\setminus s} \mu_{s'\to\alpha}(\mathbf{x}),$$

$$\lambda_{\alpha\to s}(\mathbf{x}) = \mu_{\alpha\to s}(\mathbf{x}) \sum_{s'\in\mathcal{N}(\alpha)\setminus s} \frac{\lambda_{s'\to\alpha}(\mathbf{x})}{\mu_{s'\to\alpha}(\mathbf{x})}, \qquad (2)$$

where we have defined the message $\lambda_{\alpha\to s}(\mathbf{x}) \equiv \partial_{x_\alpha}\left[\mu_{\alpha\to s}(\mathbf{x})\right]$[1]. Similarly, then for each function node $s \in S$ and neighboring variable node $\alpha \in V$ we have

$$\begin{aligned}\mu_{s\to\alpha}(\mathbf{x}) &= \partial_{x_\beta}\left[\phi_s(x_\alpha, x_\beta)\right]\mu_{\beta\to s}(\mathbf{x}) \\ &\quad + \phi_s(x_\alpha, x_\beta)\lambda_{\beta\to s}(\mathbf{x}), \\ \lambda_{s\to\alpha}(\mathbf{x}) &= \partial_{x_\alpha, x_\beta}\left[\phi_s(x_\alpha, x_\beta)\right]\mu_{\beta\to s}(\mathbf{x}) \\ &\quad + \partial_{x_\alpha}\left[\phi_s(x_\alpha, x_\beta)\right]\lambda_{\beta\to s}(\mathbf{x}), \end{aligned} \qquad (3)$$

[1]We denote all DSP messages as functions of all variables $\mathbf{x}$ in order to simplify notation, although each message is in fact only a function of variables in the subtree rooted at the current node for a tree-structured CDN.

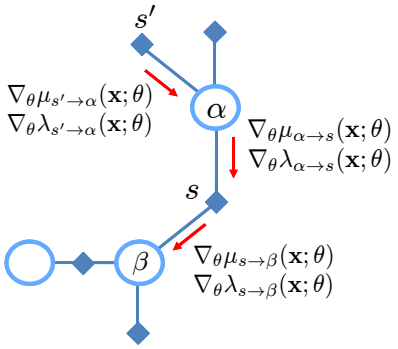


Figure 3: The gradient-derivative-product (GDP) algorithm for computing gradients and derivatives in a tree-structured CDN. Gradient messages in the GDP algorithm, which consist of the gradients of messages in the derivative-sum-product (DSP) algorithm of (Huang and Frey, 2008; Huang, 2009). At the root node α, we can compute the gradient of the negative log-likelihood by computing the gradient of the product of all incoming messages for node α.

where $\lambda_{s\to\alpha}(\mathbf{x}) = \partial_{x_\alpha}\left[\mu_{s\to\alpha}(\mathbf{x})\right]$. A complete derivation of the above algorithm for an arbitrary tree-structured CDN can be found in (Huang, 2009), where the derivation is analogous to that for the sum-product algorithm in factor graphs (Kschischang, Frey and Loeliger, 2001) but with the marginalization operator replaced by a differentiation/finite difference operator.

Now suppose the CDN is a tree-structured graph rooted at $\alpha \in V$. By applying the above algorithm starting from leaf nodes and passing messages until we reach the root node α, we can compute the mixed derivative $\partial_{\mathbf{x}}\left[F(\mathbf{x})\right]$ by multiplying together messages at α and computing its derivative as

$$\partial_{\mathbf{x}}\left[F(\mathbf{x})\right] = \partial_{x_\alpha}\left[\prod_{s\in\mathcal{N}(\alpha)} \mu_{s\to\alpha}(\mathbf{x})\right]. \qquad (4)$$

For a tree-structured CDN, the above procedure exactly yields the joint PDF/PMF $P(\mathbf{x})$. In the context of learning parametric models, this can be expressed as the likelihood function $P(\mathbf{x}|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes a vector of model parameters. In the next section, we show how the messages from the DSP algorithm can be used to derive a novel message-passing algorithm that allows us to efficiently compute gradients of the likelihood function for maximum-likelihood estimation.

3 The gradient-derivative-product algorithm for computing gradients in cumulative distribution networks

In the context of maximum-likelihood estimation with independent observations $\mathcal{D} = \{\mathbf{x}^1, \cdots, \mathbf{x}^N\}$, we minimize a loss function $\mathcal{L}(\mathcal{D};\boldsymbol{\theta})$ corresponding to the negative log-likelihood $\mathcal{L}(\mathcal{D};\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log P(\mathbf{x}^n|\boldsymbol{\theta})$ as

a function of the parameter vector $\boldsymbol{\theta}$. For a CDN, the probability model represented by the CDN corresponds to a joint CDF $F(\mathbf{x}|\boldsymbol{\theta})$ so that computing the log-likelihood corresponds to computing $P(\mathbf{x}|\boldsymbol{\theta}) = \partial_{\mathbf{x}}\Big[F(\mathbf{x}|\boldsymbol{\theta})\Big]$ and evaluating it for each observation $\mathbf{x}^n$ in $\mathcal{D}$. The above DSP algorithm can be used precisely for this purpose in tree-structured graphs: in this section we modify the messages from the DSP algorithm to compute the gradient of the negative log-likelihood $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x};\boldsymbol{\theta})$ via the introduction of additional messages. The intuition here is that much like computing derivatives of $F(\mathbf{x}|\boldsymbol{\theta})$ with respect to model variables using DSP, we can also efficiently compute gradients with respect to model parameters using message-passing.

Let the gradient of function $\phi_s(\mathbf{x}_s;\boldsymbol{\theta})$ with respect to parameter vector $\boldsymbol{\theta}$ be denoted as $\nabla_{\boldsymbol{\theta}}\phi_s(\mathbf{x}_s;\boldsymbol{\theta})$. For any function $\phi_s(\mathbf{x}_s)$ in the CDN, we define an element t of the gradient vector with respect to $\boldsymbol{\theta}$ to be zero if the function $\phi_s(\mathbf{x}_s;\boldsymbol{\theta})$ does not depend on parameter $\boldsymbol{\theta}_t$. Let the DSP messages be denoted as $\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta}), \lambda_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta}), \mu_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}), \lambda_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta})$, where the notation emphasizes the dependence of the messages on the model parameters $\boldsymbol{\theta}$. Suppose now that α is the root node for the tree-structured CDN. To compute the gradient of the negative log-likelihood $\mathcal{L}(\mathbf{x};\boldsymbol{\theta})$ for a given observation $\mathbf{x}$, we use Equation (4), which yields the gradient

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x};\boldsymbol{\theta}) &= -\nabla_{\boldsymbol{\theta}}\log\partial_{x_\alpha}\left[\prod_{s\in\mathcal{N}(\alpha)}\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})\right] \\ &= -\frac{1}{P(\mathbf{x}|\boldsymbol{\theta})}\partial_{x_\alpha}\left[\nabla_{\boldsymbol{\theta}}\prod_{s\in\mathcal{N}(\alpha)}\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})\right].\end{aligned} \tag{5}$$

Here we have applied the product rule of differential calculus to the product of messages to obtain the above expression. Given $\nabla_{\boldsymbol{\theta}}\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})$, we can then recursively apply the product rule to compute $\nabla_{\boldsymbol{\theta}}\mu_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}}\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})$ as functions of the DSP messages. For a tree-structured CDN in which each function has two argument variables, this yields

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mu_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}}\left[\prod_{s'\in\mathcal{N}(\alpha)\setminus s}\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})\right] \\ &= \mu_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta})\sum_{s'\in\mathcal{N}(\alpha)\setminus s}\frac{\nabla_{\boldsymbol{\theta}}\mu_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})}{\mu_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})},\end{aligned}$$

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}}\partial_{x_\beta}\Big[\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\Big]\mu_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}) \\ &\quad+\partial_{x_\beta}\Big[\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\Big]\nabla_{\boldsymbol{\theta}}\mu_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}) \\ &\quad+\nabla_{\boldsymbol{\theta}}\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\lambda_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}) \\ &\quad+\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\lambda_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}).\end{aligned}$$

The above expression suggests that by expanding the derivatives of the CDF in terms of DSP messages $\mu_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}), \mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})$ and their derivatives $\lambda_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}), \lambda_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})$, we can compute the gradient of the negative log-likelihood in terms of DSP messages and their gradients. By passing the gradient messages as specified above, we can compute the gradient according to Equation (5). We refer to the algorithm that computes the gradient messages in terms of DSP messages as the *gradient-derivative-product* (GDP) algorithm. The updates for the case of a CDN with functions of two arguments are given in Table 1. By computing the DSP messages and their gradients according to the GDP updates, we can obtain the gradient of the negative log-likelihood from all incoming messages at the root node α. Once computed, the gradient can be used in any gradient-based descent algorithm for maximum-likelihood. Although the above derivation assumes that one has already run the DSP algorithm to obtain the messages required for the GDP algorithm. It it worth emphasizing that the two algorithms can be run simultaneously, so that for each update one can compute DSP and GDP messages in parallel.

4 Experiments

We can now apply the GDP algorithm to problems in which a probability model is significantly easier to express as a joint CDF but we wish to estimate the model from data using a maximum-likelihood approach. In the case of heavy-tailed distributions, a multivariate probability model can be tractably specified using a CDN and estimated from data using GDP. Such distributions arise in applications such as that of modeling climate variables (Coles, 2001) and in modeling the spread of infectious disease, where we must account for dependencies between variables and for the heavy-tailed statistics of the data. A particular class of these distributions that are naturally parameterized as CDFs are max-stable processes (Coles, 2001), in which variables are modeled as the maxima over many random variables. As we demonstrate in the Appendix, CDNs allow us to tractably construct graphical models that have the properties of these max-stable processes by taking a product of local max-stable processes defined over subsets of nodes in the CDN. An example of such a distribution is the bivariate logistic distribution with Gumbel margins (Coles, 2001), given by

$$\phi_s(x,y) = \exp\Big(-\big(e^{-x/\theta_s}+e^{-y/\theta_s}\big)^{\theta_s}\Big),\ 0<\theta_s<1. \tag{6}$$

Models constructed by computing products of functions of the above type have the properties of both being heavy-tailed multivariate distributions and satisfying marginal independence constraints between vari-

- For each leaf function $\phi_s(x_{\alpha'};\boldsymbol{\theta})$, send the messages

$$\nabla_{\boldsymbol{\theta}}\mu_{s\to\alpha'}(\mathbf{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\phi_s(x_{\alpha'};\boldsymbol{\theta}),\ \nabla_{\boldsymbol{\theta}}\lambda_{s\to\alpha'}(\mathbf{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\partial_{x_{\alpha'}}\Big[\phi_s(x_{\alpha'};\boldsymbol{\theta})\Big].$$

- For each non-leaf variable α and neighboring functions $s \in \mathcal{N}(\alpha)$,

$$\nabla_{\boldsymbol{\theta}}\mu_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}) = \mu_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}) \sum_{s'\in\mathcal{N}(\alpha)\setminus s} \frac{\nabla_{\boldsymbol{\theta}}\mu_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})}{\mu_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})}$$

$$\nabla_{\boldsymbol{\theta}}\lambda_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\mu_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}) \sum_{s'\in\mathcal{N}(\alpha)\setminus s} \frac{\lambda_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})}{\mu_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})} + \mu_{\alpha\to s}(\mathbf{x};\boldsymbol{\theta}) \sum_{s'\in\mathcal{N}(\alpha)\setminus s} \frac{\nabla_{\boldsymbol{\theta}}\lambda_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})\mu_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta}) - \lambda_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\mu_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})}{\mu^2_{s'\to\alpha}(\mathbf{x};\boldsymbol{\theta})}$$

- For each non-leaf function s and neighboring variables $\alpha \in \mathcal{N}(s)$,

$$\nabla_{\boldsymbol{\theta}}\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\partial_{x_\beta}\Big[\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\Big]\mu_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}) + \partial_{x_\beta}\Big[\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\Big]\nabla_{\boldsymbol{\theta}}\mu_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\lambda_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}) + \phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\lambda_{\beta\to s}(\mathbf{x};\boldsymbol{\theta})$$

$$\nabla_{\boldsymbol{\theta}}\lambda_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\partial_{x_\alpha,x_\beta}\Big[\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\Big]\mu_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}) + \partial_{x_\alpha,x_\beta}\Big[\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\Big]\nabla_{\boldsymbol{\theta}}\mu_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\partial_{x_\alpha}\Big[\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\Big]\lambda_{\beta\to s}(\mathbf{x};\boldsymbol{\theta}) + \partial_{x_\alpha}\Big[\phi_s(x_\alpha,x_\beta;\boldsymbol{\theta})\Big]\nabla_{\boldsymbol{\theta}}\lambda_{\beta\to s}(\mathbf{x};\boldsymbol{\theta})$$

- At the root node α, compute

$$U(\mathbf{x};\boldsymbol{\theta}) = \prod_{s\in\mathcal{N}(\alpha)} \mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta}),\ \nabla_{\boldsymbol{\theta}}U(\mathbf{x};\boldsymbol{\theta}) = U(\mathbf{x};\boldsymbol{\theta}) \sum_{s\in\mathcal{N}(\alpha)} \frac{\nabla_{\boldsymbol{\theta}}\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})}{\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})},$$

$$Z(\mathbf{x};\boldsymbol{\theta}) = \sum_{s\in\mathcal{N}(\alpha)} \frac{\lambda_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})}{\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})},\ \nabla_{\boldsymbol{\theta}}Z(\mathbf{x};\boldsymbol{\theta}) = \sum_{s\in\mathcal{N}(\alpha)} \frac{\nabla_{\boldsymbol{\theta}}\lambda_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta}) - \lambda_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\mu_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})}{\mu^2_{s\to\alpha}(\mathbf{x};\boldsymbol{\theta})}$$

$$P(\mathbf{x}|\boldsymbol{\theta}) = U(\mathbf{x};\boldsymbol{\theta})Z(\mathbf{x};\boldsymbol{\theta}),\ \nabla_{\boldsymbol{\theta}}P(\mathbf{x}|\boldsymbol{\theta}) = U(\mathbf{x};\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}Z(\mathbf{x};\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}U(\mathbf{x};\boldsymbol{\theta})Z(\mathbf{x};\boldsymbol{\theta})$$

Table 1: The gradient-derivative-product (GDP) algorithm for computing the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x};\boldsymbol{\theta})$ in a tree-structured CDN in which each function is defined over two argument variables. Each message in the GDP algorithm consists of the gradient of corresponding messages in the DSP algorithm of (Huang and Frey, 2008).

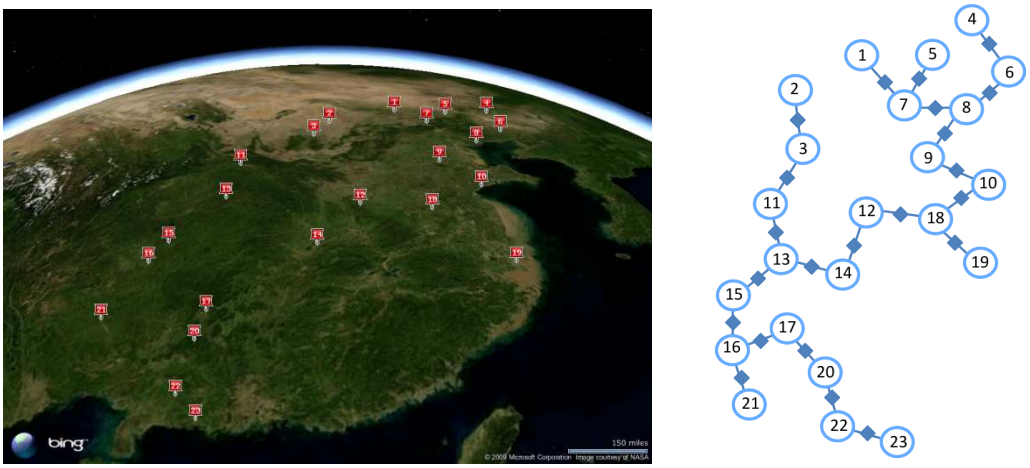
ables in the model. This makes CDNs a natural choice of models for applications in which data follow heavy-tailed distributions and in which dependencies among variables can modeled as resulting from the shared influence of latent variables. In the next sections, we will construct CDN models for rainfall and H1N1 mortality and we will use the GDP algorithm to learn such models from data.

4.1 Modeling rainfall data

For this application, the data consisted of rainfall measurements during the summer of 1998 in China (http://www.ncdc.noaa.gov/oa/gsod.html). The dataset consisted of a total of 61 daily measurements at 23 sites in China for the months of June and July 1998, during the course of which severe flooding occurred as a result of extreme deviations in rainfall. The geographic locations of the 23 sites are shown in Figure 4(a). We then constructed a CDN by choosing the approximate nearest neighboring sites for each site by geographic proximity and by then ensuring that the graph is both tree-structured and connected (Figure 4(b)). The functions in the CDN were set according to Equation (6). The CDN models both the dependencies that arise due to shared latent variables in addition to the underlying heavy-tailed statistics of the measurements. We did not perform graphical model selection here, although this would be a vital topic for future research.

Given the CDN shown in Figure 4(b), we then applied the GDP algorithm for learning the model. We used the gradients obtained from the GDP algorithm in tandem with a projected stochastic gradients optimization routine to perform learning where for each day in the training set, we compute the required gradient using GDP and perform a stochastic update. Updates which violated the constraints $0 < \theta_s < 1, s \in S$ were projected back into the feasible set by a simple thresholding.

To compare the fit of the proposed CDN model for the given data, we also fit a disconnected CDN in which



(a) (b)

Figure 4: a) The 23 sites in China from which rainfall measurements were made during the months of June, July in 1998; b) The corresponding CDN for representing the joint distribution $F(\mathbf{x})$ of recorded rainfall at each of the 23 sites.

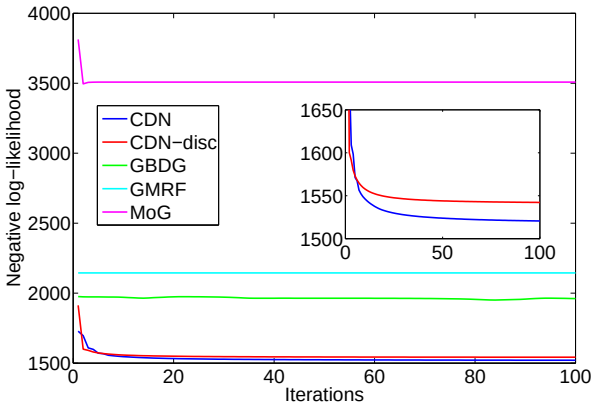


Figure 5: Negative log-likelihood for the rainfall data achieved by the CDN, disconnected CDN (CDN-disc), Gaussian bi-directed model (GBDG), Gaussian Markov random field (GMRF) and mixture of Gaussians (MoG) models as a function of number of iterations for each parameter learning algorithm.

the variables in the model are pairwise independent. We also fit Gaussian BDG and MRF models with the same graph topology as the CDN, where for the BDG model we used the iterated conditional fitting algorithm of (Drton and Richardson, 2004) and for the MRF model we used the iterative proportional scaling method of (Speed and Kiiveri, 1986). In the case of a Gaussian BDG, one learns a covariance matrix $\mathbf{\Sigma}$ subject to constraint that $(\mathbf{\Sigma})_{\alpha,\beta} = 0$ only if there is no edge connecting variable nodes α, β in the corresponding BDG. For the Gaussian MRF, we have instead constraints of the form $(\mathbf{\Sigma})^{-1}_{\alpha,\beta} = 0$. In addition to these models, we fit a mixture of Gaussians model with 2 mixture components, as we found the same model fit for mixtures with 3 or more components. Finally, we tried to apply the method of (Silva and Ghahramani, 2009), which consists of a factorial mixture of Gaussians in which each the covariance matrix $\mathbf{\Sigma}_k$ for mixture component k of the model is constrained to obey the same marginal independence constraints as those represented by the CDN and Gaussian BDG models. We found that the costs for this model, in terms of computation and memory, become intractable for bidirected graphs with more than ten variables. As a result, we were not able to obtain comparative results for this method.

The resulting training likelihoods for each of the above models as a function of the number of iterations are shown in Figure 5. We find that the parameter estimation algorithms for each of the models converge quickly, with the CDN model of Figure 4(b) achieving the best fit on the data. To appropriately gauge the predictive power of the CDN model relative to each of the models, we then performed leave-one-out cross-validation where each of the 61 days in the months of June and July was successively left out of training and we evaluate the negative log-likelihood for each model on the held-out test data. The total negative log-likelihoods for all models on test data are shown in Table 2a). We see here that the CDN is the most predictive model among those studied and that by ignoring dependencies between neighboring sites in the CDN through disconnecting the CDN, we incur poorer predictive likelihood. Furthermore, by ignoring the heavy-tailed statistics of the data, we also incur poorer predictive likelihood, as evidenced by the test performance of the models based on Gaussian distributions. Interestingly, the Gaussian BDG achieved a better test score than the Gaussian MRF, suggesting that the independence property of BDGs lead to more predictive models than the Markov property of MRFs for the

(a)

Model	$\mathcal{L}_{test}$
CDN	1.47×10^3
CDN-disc	1.52×10^3
GBDG	6.84×10^3
GMRF	1.06×10^4
MoG	1.31×10^4

(b)

Model	$\mathcal{L}_{test}$
CDN	428.19
CDN-disc	466.99
GBDG	520.99
GMRF	485.25
MoG	684.46

Table 2: Negative test log-likelihood values $\mathcal{L}_{test}$ obtained from leave-one-out cross-validation for the a) rainfall data and b) the H1N1 mortality data with CDNs from Figures 4(b),6(a). For comparison, we also computed $\mathcal{L}_{test}$ for a disconnected CDN (CDN-disc), a Gaussian bi-directed graph (GBDG) and a Gaussian Markov random field (GMRF) with the same graphical topologies as the CDNs from Figures 4(b),6(a), and a mixture of Gaussians (MoG) model.

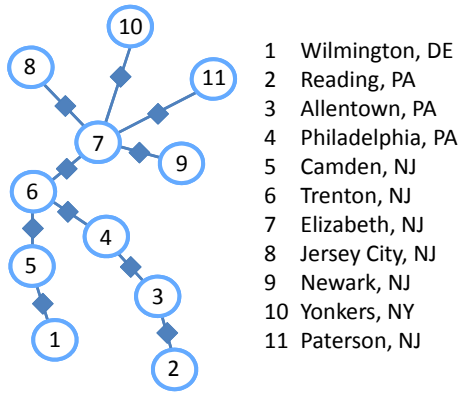
same graph topology.

4.2 Modeling H1N1 mortality rates

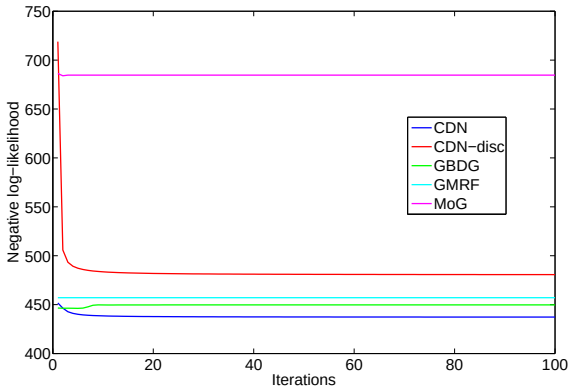
We can also apply the GDP algorithm to learning models for H1N1 data (http://finder.geocommons.com/overlays/12295) consisting of flu mortality indices for 11 cities in the Northeastern US for 29 weeks during the H1N1 epidemic of 2008-2009. We constructed a CDN based on geographic proximity of the cities (Figure 6(a)) and with CDN functions set according to Equation (6). As in the previous application, the marginal dependencies in the model were assumed to arise from the shared influence of latent variables (e.g.: degree of inter-city traffic, existence of a "patient zero"). We repeated the experiments as in the previous section for this H1N1 dataset, with resulting training and test likelihoods shown in Figure 6(b) and Table 2b). Similar to our results for the rainfall data, the CDN achieves better model fit relative to the other models, as we can account for dependencies between variables in the model and for the heavy-tailed statistics for this dataset.

5 Discussion

We have proposed a novel message-passing algorithm for maximum-likelihood learning of CDFs defined on graphs. We have applied the proposed framework to the problems of modeling rainfall and H1N1 mortality. Our results suggest additional directions for research. In particular, here we did not address the issue of graphical model selection, which would be of significant interest for the application of constructing CDNs for modelling max-stable processes. Furthermore, in practice the graphs for CDNs may contain cycles, whereas here we have designed a graph without cycles. Thus another important direction for future research would be to analyze the behavior of both the DSP and GDP algorithms for graphs with cycles and determine necessary/sufficient conditions for these algorithms to converge. Finally, there are likely to to be several applications in which multivariate heavy-tailed distributions arise, such as computer vision and finance, so that CDNs and the GDP algorithm may be applicable to problems in these fields as well.



(a)



(b)

Figure 6: a) The CDN used to model H1N1 mortality for 11 cities in the Northeastern US; b) Negative log-likelihood on H1N1 data for the CDN, disconnected CDN (CDN-disc), Gaussian bi-directed model (GBDG), Gaussian Markov random field (GMRF) and mixture of Gaussians (MoG) models as a function of number of iterations for each parameter learning algorithm.

Appendix

Formally, a stochastic process $\mathbf{X} = \{X_\alpha, \alpha \in V\}$ with joint CDF $F_{\mathbf{X}}(\mathbf{x})$ is said to be *max-stable* if for $M > 0$ independent copies of the stochastic process $\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(M)}$, the process $\mathbf{Y} = \{Y_\alpha, \alpha \in V\}$ obtained by computing $Y_\alpha = \dfrac{\max\limits_{m=1,\cdots,M} X_\alpha^{(m)} - c_\alpha^{(M)}}{d_\alpha^{(M)}}$ for constants $c_\alpha^{(M)}, d_\alpha^{(M)} > 0$ has CDF $F_{\mathbf{Y}}(\mathbf{x})$

$$F_{\mathbf{Y}}(\mathbf{x}) = F_{\mathbf{X}}(\mathbf{x}) = F_{\mathbf{X}}^M(\mathbf{D}_M\mathbf{x} + \mathbf{c}_M)$$

where $\mathbf{D}_M \in \mathbb{R}^{|V|\times|V|}, \mathbf{c}_M \in \mathbb{R}^{|V|}$ are a diagonal matrix and vector whose elements correspond to $\{c_\alpha^{(M)}, c_\alpha^{(M)}\}_{\alpha\in V}$. Intuitively, max-stable processes correspond to stochastic processes which are closed under computing the maxima of random variables in the process. The following shows that CDNs with max-stable properties can be constructed from products of max-stable processes.

Proposition 5.1. *A product of max-stable processes defined over subsets of random variables is also max-stable.*

Proof. For each variable $X_\alpha, \alpha \in V$, let $N_\alpha \equiv |\mathcal{N}(\alpha)|$ and define random variables $\{X_\alpha^{(m,t)}\}$ for $m = 1, \cdots, M$ and $s \in \mathcal{N}(\alpha)$ so that

- For any given $s \in S$ and $t \in \mathcal{N}(\alpha)$, the stochastic processes $\{X_\alpha^{(1,t)}\}_{\alpha\in\mathcal{N}(s)}, \cdots, \{X_\alpha^{(M,t)}\}_{\alpha\in\mathcal{N}(s)}$ are independent copies of the stochastic process with joint CDF $\phi_s(\mathbf{x}_s) = \mathbb{P}\left[\bigcap_{\alpha\in\mathcal{N}(s)} \left\{X_\alpha^{(m,s)} \le x_\alpha\right\}\right]$.
- For any given $m = 1, \cdots, M$, $s \in S$ and $\mathcal{N}(\alpha) = \{t^1, \cdots, t^{N_\alpha}\}$, $\{X_\alpha^{(m,t^1)}\}_{\alpha\in\mathcal{N}(s)}, \cdots, \{X_\alpha^{(m,t^{N_\alpha})}\}_{\alpha\in\mathcal{N}(s)}$ are independent copies of the stochastic process with joint CDF $\phi_s(\mathbf{x}_s) = \mathbb{P}\left[\bigcap_{\alpha\in\mathcal{N}(s)} \left\{X_\alpha^{(m,s)} \le x_\alpha\right\}\right]$.

Given the definitions for variables $X_\alpha^{(m,s)}$, let random variable Y_α be given by

$$Y_\alpha = \frac{\max_{t\in\mathcal{N}(\alpha)} \max_{m=1,\cdots,M} X_\alpha^{(m,t)} - c_\alpha^{(M,N_\alpha)}}{d_\alpha^{(M,N_\alpha)}}$$

for some $c_\alpha^{(M,N_\alpha)}, d_\alpha^{(M,N_\alpha)} > 0$. Then the joint CDF of variables $\mathbf{Y} = \{Y_\alpha\}_{\alpha\in V}$ is given by $F_\mathbf{Y}(\mathbf{x}) = \mathbb{P}\big[\bigcap_{\alpha\in V}\{Y_\alpha \le x_\alpha\}\big]$, or

$$\begin{aligned}
&\mathbb{P}\left[\bigcap_{\alpha\in V}\left\{\frac{\max_{m=1,\cdots,M}\max_{t\in\mathcal{N}(\alpha)} X_\alpha^{(m,t)} - c_\alpha^{(M,N_\alpha)}}{d_\alpha^{(M,N_\alpha)}} \le x_\alpha\right\}\right] \\
&= \mathbb{P}\left[\bigcap_{\alpha\in V}\bigcap_{t\in\mathcal{N}(\alpha)}\bigcap_{m=1}^{M}\left\{X_\alpha^{(m,t)} \le d_\alpha^{(M,N_\alpha)} x_\alpha + c_\alpha^{(M,N_\alpha)}\right\}\right] \\
&= \mathbb{P}\left[\bigcap_{s\in S}\bigcap_{\alpha\in\mathcal{N}(s)}\bigcap_{m=1}^{M}\left\{X_\alpha^{(m,s)} \le d_\alpha^{(M,N_\alpha)} x_\alpha + c_\alpha^{(M,N_\alpha)}\right\}\right] \\
&= \prod_{m=1}^{M}\prod_{s\in S}\mathbb{P}\left[\bigcap_{\alpha\in\mathcal{N}(s)}\left\{X_\alpha^{(m,s)} \le d_\alpha^{(M,N_\alpha)} x_\alpha + c_\alpha^{(M,N_\alpha)}\right\}\right] \\
&= \prod_{s\in S}\phi_s^M(\mathbf{D}_s\mathbf{x}_s + \mathbf{c}_s)
\end{aligned}$$

Now, if the functions $\phi_s(\mathbf{x}_s)$ correspond to max-stable distributions so that $\phi_s(\mathbf{x}_s) = \phi_s^M(\mathbf{D}_s\mathbf{x}_s + \mathbf{c}_s)$, then the above yields $F_\mathbf{Y}(\mathbf{x}) = F_\mathbf{X}(\mathbf{x})$.

Conversely, suppose at least one of the distributions $\phi_s(\mathbf{x}_s)$ is not max-stable. Let the set of such distributions be $\tilde{S} \subseteq S$. Then for any $\mathbf{c}, \mathbf{D}$ we can write

$$\begin{aligned}
F_\mathbf{X}^M(\mathbf{D}\mathbf{x} + \mathbf{c}) &= \prod_{s\in S}\phi_s^M(\mathbf{D}_s\mathbf{x}_s + \mathbf{c}_s) \\
&= \prod_{s\in\tilde{S}}\phi_s^M(\mathbf{D}_s\mathbf{x}_s + \mathbf{c}_s)\prod_{s\in S\setminus\tilde{S}}\phi_s(\mathbf{x}_s)
\end{aligned}$$

The above must hold for all M and in particular it must hold for $M \to \infty$. For $s \in \tilde{S}$, no choice of $\mathbf{c}, \mathbf{D}$ can allow $\phi_s(\mathbf{x}_s) = \phi_s^M(\mathbf{D}_s\mathbf{x}_s + \mathbf{c}_s)$. Since $\phi_s(\mathbf{x}_s)$ satisfies the properties of a CDF, we must have $\lim_{M\to\infty}\phi_s^M(\mathbf{D}_s\mathbf{x}_s + \mathbf{c}_s) = 0$, which violates $F_\mathbf{X}(\mathbf{x}) > 0$ and so $F_\mathbf{X}^M(\mathbf{D}\mathbf{x} + \mathbf{c}) \neq F_\mathbf{X}(\mathbf{x})$. □

References

Coles, S.G. (2001) *An introduction to statistical modelling of extreme values.* Springer.

Drton, M. and Richardson, T.S. (2004) Iterative conditional fitting for Gaussian ancestral graph models. *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI)*, 130-137.

Huang, J.C. and Frey, B.J. (2008) Cumulative distribution networks and the derivative-sum-product algorithm. *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI)*, 290-297.

Huang, J.C. and Frey, B.J. (2009) Structured ranking learning using cumulative distribution networks. *Advances in Neural Information Processing Systems (NIPS)*, **21**, 697-704.

Huang, J.C. (2009) Cumulative distribution networks: Inference, estimation and applications of graphical models for cumulative distribution functions. *Ph.D. thesis.* http://hdl.handle.net/1807/19194

Kschischang, F.R., Frey, B.J. and Loeliger, H.-A. (2001) Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory* **47(2)**, 498-519.

Silva, R. and Ghahramani, Z. (2009) Factorial mixture of Gaussians and the marginal independence model. *Proceedings of the Twelfth Annual Conference on Artificial Intelligence and Statistics (AISTATS)*, 520-527.

Speed, T.S. and Kiiveri, H.T. (1986) Gaussian Markov distributions over finite graphs. *Annals of Stat.* **14(1)**, 138-150.

Richardson, T.S. (2003) Markov properties for acyclic directed mixed graphs. *Scand. J. Statist.* **30**, 145-157.