Nonlinear Statistical Learning with Truncated Gaussian Graphical Models

Qinliang Su, Xuejun Liao, Changyou Chen, Lawrence Carin QS15, XJLIAO, CC448, LCARIN@DUKE.EDU
Duke University, Durham, NC 27519, USA

Abstract

We introduce the *truncated Gaussian graphical model (TGGM)* as a novel framework for designing statistical models for nonlinear learning. A TGGM is a Gaussian graphical model (GGM) with a subset of variables truncated to be nonnegative. The truncated variables are assumed latent and integrated out to induce a marginal model. We show that the variables in the marginal model are non-Gaussian distributed and their expected relations are nonlinear. We use expectation-maximization to break the inference of the nonlinear model into a sequence of TGGM inference problems, each of which is efficiently solved by using the properties and numerical methods of multivariate Gaussian distributions. We use the TGGM to design models for nonlinear regression and classification, with the performances of these models demonstrated on extensive benchmark datasets and compared to state-of-the-art competing results.

1. Introduction

Graphical models, which use graph-based visualization to represent statistical dependencies among random variables, have been widely used to construct multivariate statistical models (Koller & Friedman, 2009). A sophisticated model can generally represent richer statistical dependencies, but the inference can quickly become intractable as the model's complexity increases. A simple model, on the contrary, is easy to infer, but its representational power is limited.

To balance representational versatility and inferential tractability, latent variables are often added into the graphical model to obtain a tractable joint probability distribution which, when the latent variables are integrated out, induces a complicated and expressive marginal distribution over the target variables, i.e., the variables of interest (Galbraith et al., 2002). Since the complexity of these models is induced by integration, expectation-maximization (EM) (Dempster et al., 1977) can be employed to facilitate inference. The approach of EM is to break the original problem of inferring the marginal distribution into a sequence of easier problems, each of which is to infer the expected logarithmic joint distribution, where the expectation is taken over the terms of latent variables in the logarithmic domain, using the information from the previous iteration of this sequential procedure. The restricted Boltzmann machine (RBM) (Hinton, 2002) and sigmoid belief network (SBN) (Neal, 1992; Gan et al., 2015), as well as the deep networks built upon them (Salakhutdinov & Hinton, 2009; Hinton et al., 2006), are good examples of using latent variables to enhance modeling versatility while at the same time admitting tractable statistical inference.

Gaussian graphical models (GGM) constitute a subset of graphical models that have found successful application in a diverse range of areas (Honorio et al., 2009; Liu & Willsky, 2013; Oh & Deasy, 2014; Meng et al., 2014; Su & Wu, 2015a;b). The popularity of GGM may partially be attributed to the abundant applications for which the data are Gaussian distributed or approximately so, and partially attributed to the attractive properties of the multivariate Gaussian distribution which facilitate inference. However, there are many applications where the data are distributed in a way that heavily deviates from Gaussianity, and the GGM may not reveal meaningful statistical dependencies underlying the data.

What is worse, adding latent variables into a GGM does not induce enhanced marginal versatility for the target variables, as it typically does in other graphical models; this is so because the marginals of a multivariate Gaussian distribution are still Gaussian. In addition, the conditional mean of $\mathbf{y}$ given $\mathbf{x}$ is always linear in $\mathbf{x}$ whenever $(\mathbf{y}, \mathbf{x})$ are jointly Gaussian. In this sense, a GGM is inherently a linear model no matter how many latent variables are added.

To overcome the linearity of GGMs, Frey (1997), Hinton & Ghahramani (1997), and Frey & Hinton (1999) proposed to apply nonlinear transformations to Gaussian hidden variables. More recently, a deep latent Gaussian model was proposed in (Rezende et al., 2014), in which each hidden layer is connected to the output layer through a neu-

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

ral network with Gaussian noise. In these models, nonlinear transforms are applied to Gaussian variables to obtain nonlinearity at the output layer. The non-linear transforms, however, destroy the nice structure of a GGM, such as the quadratic energy function and a simple conditional dependency structure, rendering it difficult to obtain analytic learning rules and, as a result, one has to resort to less efficient sampling-based inference.

In this paper, we introduce a novel approach to inducing nonlinearity in a GGM. The new approach is simple: it adds latent variables into a GGM and truncates them below zero so that they are nonnegative. We term the resulting framework as *truncated Gaussian graphical model (TGGM)*. Although simple, the truncation leads to a remarkable result: after the truncated latent variables are integrated out, the target variables are no longer Gaussian distributed and their expected relations are no longer linear. Therefore the TGGM induces a nonlinear marginal model for the target variables, forming a striking contrast to the GGM. It should be emphasized that the approach proposed here is different from those in (Socci et al., 1998; Downs et al., 1999), which constrain the target (observed) variables to be nonnegative, without using latent variables; the nonnegative constraint in those approaches relaxes the convex function in the Gaussian distribution to a non-convex energy function that admits multimodal distributions.

The foremost advantage of the TGGM-induced nonlinear model over previous nonlinear models is the ease and efficiency with which inference can be performed. The advantage is attributed to the following two facts. First, as the nonlinear model is induced from a TGGM by integrating out the latent variables, EM can be used to break the inference into a sequence of TGGM inference problems. Second, as the truncation in a TGGM does not alter the quadratic energy function or the conditional dependency structure of the GGM, it is possible for a TGGM inference algorithm to utilize many well-studied properties of multivariate Gaussian distributions and the associated numerical methods (Johnson et al., 1994; Genz & Bretz, 2009). A second important advantage is that the conditional dependency structure of a TGGM is uniquely encoded in the precision matrix (or inverse covariance matrix) of the corresponding GGM (before the truncation is performed). By working with the precision matrix, one can conveniently design diverse structures and construct abundant types of nonlinear statistical models to fit the application at hand.

We provide several examples of leveraging the TGGM to solve machine-learning problems. In the first, we use the TGGM to construct a nonlinear regression model that can be understood as a probabilistic version of the rectified linear unit (ReLU) neural network (Glorot et al., 2011). In the second, we solve multi-class classification by using the multinomial probit link function (Albert & Chib, 1993) to transform the continuous target variables of a TGGM into categorical variables. Our main focus in the first two examples is on shallow structures, with one latent (hidden) layer of nonlinear units used in a TGGM. In the third example, we consider extensions to deep structures, by modifying the blocks in the precision matrix that are related to latent truncated variables. We use EM as the primary inference method, with the variational Bayesian (VB) approximation used for multivariate truncated Gaussian distributions. The performances of the TGGM models are demonstrated on extensive benchmark datasets and compared to state-of-the-art competing results.

2. Nonlinearity from Truncated Gaussian Graphical Models (TGGMs)

Let $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{h} \in \mathbb{R}^m$ respectively denote the target and latent variables of a TGGM, and we use $\mathbf{x}$ to denote the input variable. The TGGM is defined by the following joint probability density function

$$p(\mathbf{y}, \mathbf{h} \,|\mathbf{x}) = \mathcal{N}\left(\mathbf{y} \,\middle|\mathbf{W}_1\mathbf{h}+\mathbf{b}_1, \mathbf{P}_1^{-1}\right) \mathcal{N}_T\left(\mathbf{h} \,\middle|\mathbf{W}_0\mathbf{x}+\mathbf{b}_0, \mathbf{P}_0^{-1}\right), \quad (1)$$

where $\mathcal{N}(\mathbf{x} \,|\boldsymbol{\mu}, \mathbf{P}^{-1})$ is a multivariate Gaussian density of $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and precision matrix $\mathbf{P}$ and $\mathcal{N}_T(\mathbf{x} \,|\boldsymbol{\mu}, \mathbf{P}^{-1})$ is the associated truncated density defined as

$$\mathcal{N}_T(\mathbf{x} \,|\boldsymbol{\mu}, \mathbf{P}^{-1}) \triangleq \frac{\mathcal{N}(\mathbf{x} \,|\boldsymbol{\mu}, \mathbf{P}^{-1})\mathbb{I}(\mathbf{x} \geq \mathbf{0})}{\int_0^\infty \mathcal{N}(\mathbf{z} \,|\boldsymbol{\mu}, \mathbf{P}^{-1}) d\mathbf{z}},$$

where $\mathbb{I}(\cdot)$ is an indicator function and $\int_0^{+\infty} d\mathbf{h}$ is a simplified notation of the multiple integral. The marginal TGGM model is defined by

$$p(\mathbf{y} \,|\mathbf{x}) = \int_0^{+\infty} p(\mathbf{y}, \mathbf{h} \,|\mathbf{x}) d\mathbf{h}. \quad (2)$$

To see how the truncation $\mathbf{h} \geq \mathbf{0}$ affects the marginal TGGM $p(\mathbf{y} \,|\mathbf{x})$, we rewrite (1) equivalently as

$$p(\mathbf{y}, \mathbf{h} \,|\mathbf{x}) = \frac{\mathcal{N}(\mathbf{h}|\boldsymbol{\mu}_{\mathbf{h}|\mathbf{x},\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{h}|\mathbf{x},\mathbf{y}}) \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_{\mathbf{y}|\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{y}|\mathbf{x}}) \mathbb{I}(\mathbf{h} \geq \mathbf{0})}{\int_0^{+\infty} \mathcal{N}\left(\mathbf{z} \,\middle|\mathbf{W}_0\mathbf{x}+\mathbf{b}_0, \mathbf{P}_0^{-1}\right) d\mathbf{z}}, \quad (3)$$

where $\boldsymbol{\mu}_{\mathbf{y}|\mathbf{x}} = \mathbf{W}_1(\mathbf{W}_0\mathbf{x} + \mathbf{b}_0) + \mathbf{b}_1$, $\boldsymbol{\Sigma}_{\mathbf{y}|\mathbf{x}} = \mathbf{W}_1\mathbf{P}_0^{-1}\mathbf{W}_1 + \mathbf{P}_1^{-1}$, $\boldsymbol{\mu}_{\mathbf{h}|\mathbf{x},\mathbf{y}} = (\mathbf{P}_0 + \mathbf{W}_1^T\mathbf{P}_1\mathbf{W}_1)^{-1}(\mathbf{P}_0(\mathbf{W}_0\mathbf{x} + \mathbf{b}_0) + \mathbf{W}_1^T\mathbf{P}_1(\mathbf{y} - \mathbf{b}_1))$, and $\boldsymbol{\Sigma}_{\mathbf{h}|\mathbf{x},\mathbf{y}} = (\mathbf{P}_0 + \mathbf{W}_1^T\mathbf{P}_1\mathbf{W}_1)^{-1}$. From (3) and (2) follows

$$p(\mathbf{y} \,|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_{\mathbf{y}|\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{y}|\mathbf{x}}) \frac{\int_0^{+\infty} \mathcal{N}(\mathbf{h}|\boldsymbol{\mu}_{\mathbf{h}|\mathbf{x},\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{h}|\mathbf{x},\mathbf{y}}) d\mathbf{h}}{\int_0^{+\infty} \mathcal{N}\left(\mathbf{h} \,\middle|\mathbf{W}_0\mathbf{x}+\mathbf{b}_0, \mathbf{P}_0^{-1}\right) d\mathbf{h}}. \quad (4)$$

It is seen from (4) that the target distribution induced by a TGGM consists of two multiplicative terms. The first term

is a Gaussian distribution induced by the associated GGM (for which $\mathbf{h}$ is not truncated). The second term modulates the Gaussian term into a more complicated and non-Gaussian distribution. As an example, one can verify that (4) is a skewed normal with $w_0 = b_0 = b_1 = 0$, $p_0 = 1$, $p_1 = 2$, and $w_1 = 1/2$ (Mudholkar & Hutson, 2000).

The modeling versatility of a TGGM is primarily influenced by m, the number of truncated latent variables, and $\mathbf{P}_0$, which encodes marginal dependencies of these variables. With a proper choice of m and $\mathbf{P}_0$, one can construct TGGM models to solve diverse nonlinear learning tasks. The nonlinearity induced by a TGGM is seen from the expression of $\mathbb{E}[\mathbf{y}|\mathbf{x}]$ which, using (1), is found to be

$$\mathbb{E}[\mathbf{y}|\mathbf{x}] = \mathbf{W}_1\mathbb{E}[\mathbf{h}|\mathbf{x}] + \mathbf{b}_1, \tag{5}$$

where $\mathbb{E}[\mathbf{h}|\mathbf{x}]$ is the expectation with respect to $\mathcal{N}_T\left(\mathbf{h}\left|\mathbf{W}_0\mathbf{x}+\mathbf{b}_0, \mathbf{P}_0^{-1}\right.\right)$. Due to the truncation $\mathbf{h} \geq 0$, the expectation $\mathbb{E}[\mathbf{h}|\mathbf{x}]$ is a nonlinear function of $\mathbf{x}$. By contrast, if $\mathbf{h}$ is not truncated, one has $\mathbb{E}[\mathbf{h}|\mathbf{x}] = \mathbf{W}_0\mathbf{x} + \mathbf{b}_0$, which is a linear function of $\mathbf{x}$. Thus, a TGGM induces nonlinearity through the truncation of its latent variables.

The nonlinearity can be controlled by adjusting $\mathbf{P}_0$. For example, if we set $\mathbf{P}_0 = \frac{1}{\sigma^2}\mathbf{I}_m$, where $\mathbf{I}_m$ is a $m \times m$ identity matrix, we obtain

$$\mathbb{E}[\mathbf{h}(k)|\mathbf{x}] = g\left(\mathbf{W}_0(k,:)\mathbf{x} + \mathbf{b}_0(k), \sigma\right), \tag{6}$$

where $\mathbf{h}(k)$ is the k-th element of $\mathbf{h}$ and $\mathbf{W}_0(k,:)$ the k-th row of $\mathbf{W}_0$ using Matlab notations, and $g(\mu,\sigma)$ is the mean of the univariate truncated normal distribution $\mathcal{N}_T(x\left|\mu,\sigma^2\right.)$. The formula of $g(\mu,\sigma)$ is given by (Johnson et al., 1994)

$$g(\mu,\sigma) \triangleq \mu + \sigma\frac{\phi\left(\frac{\mu}{\sigma}\right)}{\Phi\left(\frac{\mu}{\sigma}\right)}, \tag{7}$$

where $\phi(z) \triangleq \frac{1}{\sqrt{2\pi}}\exp^{-\frac{z^2}{2}}$ is the probability density function (PDF) of the standard normal distribution, and $\Phi(z) \triangleq \int_{-\infty}^{z}\phi(t)dt$ its cumulative distribution function (CDF). Figure 1 shows $g(\mu,\sigma)$ as a function of μ, for various values of σ, alongside $\max(0,\mu)$, which is the activation function used in ReLU neural networks (Glorot et al., 2011). It is seen that $g(\mu,\sigma)$ is a soft version of $\max(0,\mu)$ and $\lim_{\sigma\to 0} g(\mu,\sigma) = \max(0,\mu)$.

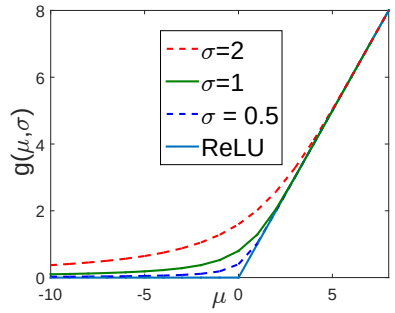

Figure 1. Visualization of $g(\mu,\sigma)$ as a function of μ, for various values of σ, in comparison to $\max(0,\mu)$.

3. Nonlinear Regression with TGGMs

We begin with a nonlinear regression model constructed from a simple TGGM, in which we restrict $\mathbf{P}_0$ and $\mathbf{P}_1$ to diagonal matrices: $\mathbf{P}_0 = \sigma_0^2\mathbf{I}_m$ and $\mathbf{P}_1 = \sigma_1^2\mathbf{I}_n$. By (6)-(7) and the arguments there, the $\mathbb{E}(\mathbf{y}|\mathbf{x})$ in such a TGGM implements the output of a soft-version ReLU neural network, which has a single layer of m hidden units with the activation function $g(\cdot,\sigma)$, and uses $\mathbf{W}_0$ and $\mathbf{W}_1$ as the input-to-hidden and hidden-to-output weights, respectively.

3.1. Maximum-Likelihood (ML) Parameter Estimation

Given a training dataset consisting of the inputs (covariates) $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$ and the outputs (responses) $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N]$, the log-likelihood function is

$$\mathcal{L}(\mathbf{\Theta}) \triangleq \ln\int_0^\infty p(\mathbf{Y},\mathbf{H}|\mathbf{X};\mathbf{\Theta})d\mathbf{H} \tag{8}$$

where $\mathbf{\Theta} \triangleq \{\mathbf{W}_0, \mathbf{W}_1, \mathbf{b}_1, \mathbf{b}_0\}$, and

$$\begin{aligned} p(\mathbf{Y},\mathbf{H}\,|\mathbf{X},\mathbf{\Theta}) = \textstyle\prod_{i=1}^N \mathcal{N}_T\left(\mathbf{h}_i\left|\mathbf{W}_0\mathbf{x}_i + \mathbf{b}_0, \sigma_0^2\mathbf{I}_m\right.\right) \\ \times\mathcal{N}\left(\mathbf{y}_i\left|\mathbf{W}_1\mathbf{h}_i + \mathbf{b}_1, \sigma_1^2\mathbf{I}_n\right.\right). \end{aligned} \tag{9}$$

Let $q(\mathbf{H}|\widetilde{\mathbf{\Theta}})$ be an arbitrary PDF with parameters $\widetilde{\mathbf{\Theta}}$, defined on $\{\mathbf{H} : \mathbf{H} \geq 0\}$. It follows from (8)

$$\begin{aligned} \mathcal{L}(\mathbf{\Theta}) &= \ln\int_0^\infty q(\mathbf{H}|\widetilde{\mathbf{\Theta}})\frac{p(\mathbf{Y},\mathbf{H}|\mathbf{X};\mathbf{\Theta})}{q(\mathbf{H}|\widetilde{\mathbf{\Theta}})}d\mathbf{H}, \\ &\geq \int_0^\infty q(\mathbf{H}|\widetilde{\mathbf{\Theta}})\ln\frac{p(\mathbf{Y},\mathbf{H}|\mathbf{X};\mathbf{\Theta})}{q(\mathbf{H}|\widetilde{\mathbf{\Theta}})}d\mathbf{H}, \\ &\quad \text{(Jensen's Inequality)} \\ &= \mathcal{L}(\mathbf{\Theta}) - \mathrm{KL}(q(\cdot|\widetilde{\mathbf{\Theta}})||p(\cdot|\mathbf{Y},\mathbf{X};\mathbf{\Theta})), \\ &\triangleq \mathcal{Q}_{q(\cdot|\widetilde{\mathbf{\Theta}})}(\mathbf{\Theta}). \end{aligned} \tag{10}$$

where $\mathrm{KL}(q(\cdot)||p(\cdot))$ denotes the Kullback-Leibler (KL) distance, and, if $\exists\ \mathbf{H}$ such that $q(\mathbf{H}) = 0$, the limit values are used to lead to $\frac{q(\mathbf{H})}{q(\mathbf{H})} = 1$ and $q(\mathbf{H})\ln q(\mathbf{H}) = 0$. In general $q(\cdot)$ is parameterized differently from the TGGM; when $q(\mathbf{H}|\widetilde{\mathbf{\Theta}}) = p(\mathbf{H}|\mathbf{Y},\mathbf{X},\mathbf{\Theta})$, however, we let $q(\cdot)$ use the same parameterization as the TGGM so that $\widetilde{\mathbf{\Theta}} = \mathbf{\Theta}$. In this case, we drop the subscript to simply write $\mathcal{Q}(\mathbf{\Theta}|\widetilde{\mathbf{\Theta}}) \equiv \mathcal{Q}_{p(\cdot|\mathbf{Y},\mathbf{X},\widetilde{\mathbf{\Theta}})}(\mathbf{\Theta})$, which, by (10), can be further written as

$$\mathcal{Q}(\mathbf{\Theta}|\widetilde{\mathbf{\Theta}}) = \mathcal{L}(\mathbf{\Theta}) - \mathrm{KL}(p(\cdot|\mathbf{Y},\mathbf{X};\widetilde{\mathbf{\Theta}})||p(\cdot|\mathbf{Y},\mathbf{X};\mathbf{\Theta})), \tag{11}$$

From (11) follows the EM algorithm. First, it is clear that $\mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}) = \mathcal{L}(\boldsymbol{\Theta})$. Thus, for a sequence $\{\boldsymbol{\Theta}_t\}$ satisfying

$$\boldsymbol{\Theta}_{t+1} = \arg\max_{\boldsymbol{\Theta}} \mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}_t), \tag{12}$$

one deduces $\mathcal{L}(\boldsymbol{\Theta}_t) = \mathcal{Q}(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}_t) \leq \mathcal{Q}(\boldsymbol{\Theta}_{t+1}|\boldsymbol{\Theta}_t) \leq \mathcal{L}(\boldsymbol{\Theta}_{t+1})$, where the last inequality follows from (11). By successively solving (12), starting from initial $\boldsymbol{\Theta}_1$, the EM algorithm produces a sequence $\{\boldsymbol{\Theta}_t : t \geq 1\}$ that monotonically increases $\mathcal{L}(\boldsymbol{\Theta}_t)$. To ensure $\mathcal{L}(\boldsymbol{\Theta}_{t+1}) > \mathcal{L}(\boldsymbol{\Theta}_t)$, one only requires $\mathcal{Q}(\boldsymbol{\Theta}_{t+1}|\boldsymbol{\Theta}_t) > \mathcal{Q}(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}_t)$. Therefore it is not necessary to solve (12) completely; rather it is sufficient to perform a single-step gradient ascent from $\boldsymbol{\Theta}_t$,

$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t + \gamma_t \nabla_{\boldsymbol{\Theta}} \mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}_t)\big|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}_t}, \tag{13}$$

where $\mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}_t) = \int_0^\infty p(\mathbf{H}|\mathbf{Y},\mathbf{X};\boldsymbol{\Theta}_t) \ln p(\mathbf{Y},\mathbf{H}|\mathbf{X};\boldsymbol{\Theta}) d\mathbf{Z}$. To find the gradient, it is helpful to write $p(\mathbf{Y},\mathbf{H}|\mathbf{X};\boldsymbol{\Theta}) = \frac{1}{Z(\mathbf{X};\boldsymbol{\Theta})} e^{-E(\mathbf{Y},\mathbf{H}|\mathbf{X};\boldsymbol{\Theta})}$, where $E = \sum_{i=1}^N \frac{\|\mathbf{h}_i - \mathbf{W}_0\mathbf{x}_i\|^2}{2\sigma_0^2} + \sum_{i=1}^N \frac{\|\mathbf{y}_i - \mathbf{W}_1\mathbf{h}_i\|^2}{\sigma_1^2}$ is the energy function and Z the normalization. The gradient can then be expressed as

$$\nabla_{\boldsymbol{\Theta}} \mathcal{Q} = -\mathbb{E}\left[\frac{\partial E}{\partial \boldsymbol{\Theta}}\middle|\mathbf{Y},\mathbf{X}\right] + \mathbb{E}\left[\frac{\partial E}{\partial \boldsymbol{\Theta}}\middle|\mathbf{X}\right], \tag{14}$$

where $\mathbb{E}[\cdot|\mathbf{Y},\mathbf{X}]$ denotes the expectation with respect to (w.r.t.) $p(\mathbf{H}|\mathbf{X},\mathbf{Y};\boldsymbol{\Theta}_t)$, and $\mathbb{E}[\cdot|\mathbf{X}]$ the expectation w.r.t. $p(\mathbf{Y},\mathbf{H}|\mathbf{X};\boldsymbol{\Theta}_t) = p(\mathbf{Y}|\mathbf{H};\boldsymbol{\Theta}_t)\, p(\mathbf{H}|\mathbf{X};\boldsymbol{\Theta}_t)$. Specifically, the partial derivatives of $\mathcal{Q}$ w.r.t. $\mathbf{W}_0$ and $\mathbf{W}_1$ can respectively be derived as

$$\frac{\partial \mathcal{Q}}{\partial \mathbf{W}_0} = -\frac{1}{\sigma_0^2}\left(\mathbb{E}[\mathbf{H}|\mathbf{X}] - \mathbb{E}[\mathbf{H}|\mathbf{Y},\mathbf{X}]\right)\mathbf{X}^T, \tag{15}$$

$$\frac{\partial \mathcal{Q}}{\partial \mathbf{W}_1} = -\frac{1}{\sigma_1^2}\Big(\mathbf{W}_1\mathbb{E}\big[\mathbf{H}\mathbf{H}^T|\mathbf{Y},\mathbf{X}\big] - (\mathbf{Y} - \mathbf{b}_1\mathbf{1}_N^T)\mathbb{E}[\mathbf{H}^T|\mathbf{Y},\mathbf{X}]\Big), \tag{16}$$

where $\mathbf{1}_N$ is a column vector of ones. The derivatives w.r.t. $\mathbf{b}_0$ and $\mathbf{b}_1$ can be derived similarly.

3.2. ML Estimation versus Backpropagation

As mentioned earlier, for a TGGM with diagonal $\mathbf{P}_0$ and $\mathbf{P}_1$, $\mathbb{E}(\mathbf{y}|\mathbf{x})$ implements the output of a soft-version ReLU neural network that use (7) as the activation function at each hidden unit. This suggests one can use backpropagation (BP) to minimize the error between $\mathbb{E}(\mathbf{Y}|\mathbf{X})$ and the training samples of $\mathbf{Y}$, as one does in training a standard ReLU network (Glorot et al., 2011).

A popular choice of the error function used by BP is the squared error which, in the case here, is given by $\mathcal{E} \triangleq \frac{1}{2\sigma_1^2}\left\|\mathbf{W}_1\mathbb{E}(\mathbf{H}|\mathbf{X}) + \mathbf{b}_1\mathbf{1}_N^T - \mathbf{Y}\right\|^2$. Minimization of the squared error is equivalent to maximization of the likelihood under the assumption that $\mathbf{y}|\mathbf{x} \sim \mathcal{N}(\mathbf{y}|\mathbb{E}(\mathbf{y}|\mathbf{x}), \sigma_1^2)$. However, we have shown in (4) that $p(\mathbf{y}|\mathbf{x})$ is a non-Gaussian distribution. Therefore, BP does not maximize the likelihood of the TGGM in the rigorous sense.

To gain a deeper understanding of the relation between BP and ML estimation, we analyze the update equations of BP and compare them to those of the ML estimator. The BP performs gradient descent of the squared error, with the required partial derivatives given by

$$\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0} = -\left(\left(\mathbf{W}_1^T(\mathbb{E}(\mathbf{Y}|\mathbf{X}) - \mathbf{Y})\right) \odot \frac{\mathrm{Var}(\mathbf{H}|\mathbf{X})}{\sigma_0^2}\right)\mathbf{X}^T, \tag{17}$$

$$\frac{\partial \mathcal{E}}{\partial \mathbf{W}_1} = -\frac{1}{\sigma_1^2}\Big(\mathbf{W}_1\mathbb{E}(\mathbf{H}|\mathbf{X})\,\mathbb{E}(\mathbf{H}^T|\mathbf{X}) - (\mathbf{Y} - \mathbf{b}_1\mathbf{1}_N^T)\mathbb{E}(\mathbf{H}^T|\mathbf{X})\Big), \tag{18}$$

where $\odot$ is the Hadamard product and $\mathrm{Var}(\mathbf{H}|\mathbf{X}) \triangleq \mathbb{E}\Big[\big(\mathbf{H} - \mathbb{E}(\mathbf{H}|\mathbf{X})\big) \odot \big(\mathbf{H} - \mathbb{E}(\mathbf{H}|\mathbf{X})\big)\Big|\mathbf{X}\Big]$ is a matrix of variances. Comparing (18) to (16), we can see that the direction of $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_1}$ is an approximation to that of $\frac{\partial \mathcal{Q}}{\partial \mathbf{W}_1} = -\frac{1}{\sigma_1^2}\big(\mathbf{W}_1\mathbb{E}\big[\mathbf{H}\mathbf{H}^T|\mathbf{Y},\mathbf{X}\big] - (\mathbf{Y} - \mathbf{b}_1\mathbf{1}_N^T)\mathbb{E}[\mathbf{H}^T|\mathbf{Y},\mathbf{X}]\big)$ by replacing the posterior expectations $\mathbb{E}[\mathbf{H}\mathbf{H}^T|\mathbf{Y},\mathbf{X}]$ and $\mathbb{E}[\mathbf{H}|\mathbf{Y},\mathbf{X}]$ with the corresponding prior expectations $\mathbb{E}[\mathbf{H}|\mathbf{X}]\mathbb{E}[\mathbf{H}^T|\mathbf{X}]$ and $\mathbb{E}[\mathbf{H}|\mathbf{X}]$. Hence, the ML estimator makes a more sufficient use of the available information, in the sense that it takes $\mathbf{Y}$ into account while BP does not.

To relate $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0}$ to $\frac{\partial \mathcal{Q}}{\partial \mathbf{W}_0}$, we require the lemma below.

Lemma 1. *Let $\mathbf{U}$ be a matrix of random numbers with $\mathbf{U}(:,j) \sim \mathcal{N}(\mathbb{E}[\mathbf{H}(:,j)|\mathbf{X}(:,j)], \rho^2)$. If $\mathbf{Y}$ are generated according to $\mathbf{y}_j|\mathbf{U}(:,j) \sim \mathcal{N}(\mathbf{y}_j|\mathbf{W}_1\mathbf{U}(:,j) + \mathbf{b}_1, \sigma_1^2\mathbf{I})$, $\forall j$, then the $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0}$ in (17) can be equivalently expressed as*

$$\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0} = -\frac{1}{\rho^2}\Big[\left(\left(\sigma_1^2\mathbf{I} + \rho^2\mathbf{W}_1^T\mathbf{W}_1\right)\left(\mathbb{E}[\mathbf{H}|\mathbf{X}] - \mathbb{E}[\mathbf{U}|\mathbf{Y},\mathbf{X}]\right)\right) \odot \mathrm{Var}(\mathbf{H}|\mathbf{X})/\sigma_0^2\Big]\mathbf{X}^T. \tag{19}$$

Proof. Since the prior $p(\mathbf{U})$ and the conditional $p(\mathbf{Y}|\mathbf{U})$ are both Gaussian, the joint distribution $p(\mathbf{Y},\mathbf{U})$ is also Gaussian. As a result, the posterior $p(\mathbf{U}|\mathbf{Y})$ is a Gaussian distribution with the mean given by $\mathbb{E}[\mathbf{U}|\mathbf{Y},\mathbf{X}] = \left((\sigma_1^2/\rho^2)\mathbf{I} + \mathbf{W}_1^T\mathbf{W}_1\right)^{-1}\mathbf{W}_1^T(\mathbf{Y} - \mathbb{E}[\mathbf{Y}|\mathbf{X}]) + \mathbb{E}[\mathbf{H}|\mathbf{X}]$. It then follows that $\mathbf{W}_1^T(\mathbf{Y} - \mathbb{E}[\mathbf{Y}|\mathbf{X}]) = \left((\sigma_1^2/\rho^2)\mathbf{I} + \mathbf{W}_1^T\mathbf{W}_1\right)(\mathbb{E}[\mathbf{U}|\mathbf{Y},\mathbf{X}] - \mathbb{E}[\mathbf{H}|\mathbf{X}])$, which is substituted into (17) to yield (19). □

As (19) holds for any $\rho^2 > 0$, it is also true when $\rho^2 \approx 0$, in which case the value of ρ^2 has little influence on the direction of $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0}$; Therefore, we can make ρ^2 sufficiently small such that $\left(\sigma_1^2\mathbf{I} + \rho^2\mathbf{W}_1^T\mathbf{W}_1\right) \approx \sigma_1^2\mathbf{I}$, and consequently $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0} \approx -\frac{\sigma_1^2}{\sigma_0^2\rho^2}\Big[\left(\left(\mathbb{E}[\mathbf{H}|\mathbf{X}] - \mathbb{E}[\mathbf{U}|\mathbf{Y},\mathbf{X}]\right)\right) \odot$

$\mathrm{Var}(\mathbf{H}|\mathbf{X})\Big]\mathbf{X}^T$. Comparing the latter equation to (15), we see that the gradients $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0}$ and $\frac{\partial \mathcal{Q}}{\partial \mathbf{W}_0}$ are different in three aspects: (*i*) the $\mathbb{E}[\mathbf{H}|\mathbf{Y},\mathbf{X}]$ in $\frac{\partial \mathcal{Q}}{\partial \mathbf{W}_0}$ is replaced by $\mathbb{E}[\mathbf{U}|\mathbf{Y},\mathbf{X}]$ in $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0}$; (*ii*) a new factor $\mathrm{Var}(\mathbf{H}|\mathbf{X})$ arises in $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0}$; and (*iii*) the multiplicative constants are different. Since (*iii*) has no influence on the directions of the gradients, we focus on (*i*) and (*ii*). The new factor $\mathrm{Var}(\mathbf{H}|\mathbf{X})$ in $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0}$ does not depend on $\mathbf{Y}$, so it plays no direct role in back-propagating the information from the output layer to the input layer. The only term of $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0}$ that contains $\mathbf{Y}$ is $\mathbb{E}[\mathbf{U}|\mathbf{Y},\mathbf{X}]$, which plays the primary and direct role in sending back the information from the output layer when updating the input-to-hidden weights $\mathbf{W}_0$. Since $\mathbb{E}[\mathbf{U}|\mathbf{Y},\mathbf{X}]$ is obtained under the assumption that $\mathbf{Y}$ is generated from Gaussian latent variables $\mathbf{U}$, it is clear that the gradient $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_0}$ used by BP does not fully reflect the underlying truncated characteristics of $\mathbf{H}$ in the TGGM model. On the contrary, $\mathbb{E}[\mathbf{H}|\mathbf{Y},\mathbf{X}]$ is the true posterior mean of $\mathbf{H}$ under the truncation assumption.

In summary, BP uses update rules that are closely related to those of the ML estimator, but it does not fully exploit the available information in updating the TGGM parameters. In particular, BP ignores $\mathbf{Y}$ when it uses $\mathbb{E}(\mathbf{H}|\mathbf{X})$, instead of $\mathbb{E}(\mathbf{H}|\mathbf{Y},\mathbf{X})$, to update $\mathbf{W}_1$; it makes an incorrect assumption about the latent variables when it uses $\mathbb{E}(\mathbf{U}|\mathbf{Y},\mathbf{X})$, instead of $\mathbb{E}(\mathbf{H}|\mathbf{Y},\mathbf{X})$, to update $\mathbf{W}_0$. These somewhat defective update equations are attributed to the fact that BP makes a wrong assumption from the very beginning, i.e., BP assumes $p(\mathbf{y}|\mathbf{x})$ is a Gaussian distribution while the distribution is truly non-Gaussian as shown in (4). For these reasons, BP usually produces worse learning results for a TGGM than the ML estimator, and this will be discussed further in the experiments.

3.3. Technical Details

A key step of the ML estimator is calculation of the prior and posterior expectations $\mathbb{E}[\cdot|\mathbf{X}]$ and $\mathbb{E}[\cdot|\mathbf{Y},\mathbf{X}]$ in (15) and (16). Since $\mathbf{P}_0 = \sigma_0^2\mathbf{I}_m$ is diagonal, the components in $\mathbf{h}_i|\mathbf{x}_i$ are independent; further, the training samples are assumed independent to each other. Therefore $p(\mathbf{H}|\mathbf{X})$ factorizes into a product of univariate truncated normal densities, $p(\mathbf{H}|\mathbf{X}) = \prod_{i=1}^N \prod_{k=1}^m \mathcal{N}_T(\mathbf{h}_i(k)|\mathbf{W}_0(k,:)\mathbf{x}_i, \sigma_0^2)$, where each univariate density is associated with a single truncated variable and a particular training sample. Each of these densities has its mean and variance given by $\mathbb{E}[\mathbf{h}_i(k)|\mathbf{x}_i] = g(\mathbf{W}_0(k,:)\mathbf{x}_i + \mathbf{b_0}(k), \sigma_0^2)$ and $\mathrm{Var}[\mathbf{h}_i(k)|\mathbf{x}_i] = \omega^2\left(\mathbf{W}_0(k,:)\mathbf{x}_i + \mathbf{b_0}(k), \sigma_0^2\right)$, respectively, where $g(\cdot,\cdot)$ is defined in (7) , and

$$\omega^2(\mu,\sigma) \triangleq \sigma^2\left(1 - \frac{\mu}{\sigma}\frac{\phi\left(\frac{\mu}{\sigma}\right)}{\Phi\left(\frac{\mu}{\sigma}\right)} - \frac{\phi^2\left(\frac{\mu}{\sigma}\right)}{\Phi^2\left(\frac{\mu}{\sigma}\right)}\right) \quad (20)$$

is the variance of the truncated normal $\mathcal{N}_T(z|\mu,\sigma^2)$ (Johnson et al., 1994). Due to the independences, one can easily compute $\mathbb{E}\left[\mathbf{H}\mathbf{H}^T|\mathbf{X}\right] = \sum_{i=1}^N \mathbb{E}\left[\mathbf{h}_i\mathbf{h}_i^T|\mathbf{x}_i\right]$, with $\mathbb{E}\left[\mathbf{h}_i\mathbf{h}_i^T|\mathbf{x}_i\right] = \mathbb{E}[\mathbf{h}_i|\mathbf{x}_i]\mathbb{E}[\mathbf{h}_i^T|\mathbf{x}_i] + \mathrm{diag}\left(\mathrm{Var}[\mathbf{h}_i|\mathbf{x}_i]\right)$.

For the posterior expectation $\mathbb{E}[\cdot|\mathbf{Y},\mathbf{X}]$, it could be computed by means of numerical integration. Multivariate integrations in normal distributions have been well studied and many effective algorithms have been developed (Genz, 1992; Genz & Bretz, 2009). Another approach is to use the mean-field variational Bayesian (VB) method (Jordan et al., 1999; Corduneanu & Bishop, 2001), which approximates the true posterior $p(\mathbf{H}|\mathbf{Y},\mathbf{X})$ with a factorized distribution $q(\mathbf{H}|\widetilde{\mathbf{\Theta}}) = \prod_{i=1}^N \prod_{k=1}^m q(\mathbf{h}_i(k)|\widetilde{\mathbf{\Theta}})$, parameterized by $\widetilde{\mathbf{\Theta}}$. The approximate posterior is found by minimizing $\mathrm{KL}\left(q(\mathbf{H}|\widetilde{\mathbf{\Theta}})||p(\mathbf{H}|\mathbf{Y},\mathbf{X};\mathbf{\Theta})\right)$, or maximizing the lower bound $\mathcal{Q}_{q(\cdot|\widetilde{\mathbf{\Theta}})}(\mathbf{\Theta})$, as shown in (10).

As $\mathbf{h}_i$ is independent of $\mathbf{h}_j$, $\forall\ i,j$, given $\mathbf{Y}$ and $\mathbf{X}$, the KL distance can be equivalently expressed as $\sum_{i=1}^N \mathrm{KL}\left(q(\mathbf{h}_i|\widetilde{\mathbf{\Theta}})||p(\mathbf{h}_i|\mathbf{Y},\mathbf{X};\mathbf{\Theta})\right)$ and each term in the sum can be minimized independently. Given $\{q(\mathbf{h}_j(\ell)) : \ell \neq k\}$, the ith term of the KL distance is minimized by

$$q(\mathbf{h}_i(k)|\widetilde{\mathbf{\Theta}}) \propto e^{\langle \ln p(\mathbf{y}_i,\mathbf{h}_i|\mathbf{x}_i)\rangle_{-k}}, \quad (21)$$

where $p(\mathbf{y}_i,\mathbf{h}_i|\mathbf{x}_i) = \mathcal{N}_T\left(\mathbf{h}_i\left|\mathbf{W}_0\mathbf{x}_i + \mathbf{b}_0, \sigma_0^2\mathbf{I}_m\right.\right) \times \mathcal{N}\left(\mathbf{y}_i\left|\mathbf{W}_1\mathbf{h}_i + \mathbf{b}_1, \sigma_1^2\mathbf{I}_n\right.\right)$ and $\langle\cdot\rangle_{-k}$ denotes the expectation w.r.t. $\prod_{\ell\neq k} q(\mathbf{h}_j(\ell))$. From (21), one obtains

$$q\left(\mathbf{h}_i(k)|\widetilde{\mathbf{\Theta}}\right) = \mathcal{N}_T\left(\mathbf{h}_i(k)\left|\boldsymbol{\xi}_i(k), \frac{1}{\mathbf{P}(k,k)}\right.\right), \quad (22)$$

where $\mathbf{P} \triangleq \frac{1}{\sigma_0^2}\mathbf{I}_m + \frac{1}{\sigma_1^2}\mathbf{W}_1^T\mathbf{W}_1$, $\boldsymbol{\xi}_i$ is a vector with its k-th element defined as $\boldsymbol{\xi}_i(k) = \frac{\boldsymbol{\gamma}_i(k) - \tilde{\mathbf{P}}(k,-k)\langle \mathbf{h}_i(-k)\rangle_{-k}}{\mathbf{P}(k,k)}$, $\boldsymbol{\gamma}_i \triangleq \frac{1}{\sigma_0^2}\left(\mathbf{W}_0\mathbf{x}_i + \mathbf{b}_0\right) + \frac{1}{\sigma_1^2}\mathbf{W}_1^T(\mathbf{y}_i - \mathbf{b}_1)$, $\tilde{\mathbf{P}} \triangleq \mathbf{P} - \mathrm{diag}(\mathbf{P})$, $\mathbf{P}(k,-k)$ is the k-th row of $\mathbf{P}$ with its k-th element deleted, and $\mathbf{h}_i(-k)$ is the subvector of $\mathbf{h}_i$ missing the k-th element.

The KL distance $\mathrm{KL}\left(q(\mathbf{h}_i|\widetilde{\mathbf{\Theta}})||p(\mathbf{h}_i|\mathbf{Y},\mathbf{X};\mathbf{\Theta})\right)$ monotonically decreases as one cyclically computes (22) through $k = 1, 2, \cdots, m$. One shall perform enough cycles until the KL distance converges. Upon convergence, $q(\mathbf{h}_i(k)|\widetilde{\mathbf{\Theta}})$ is used as the best approximation to $p(\mathbf{h}_i|\mathbf{Y},\mathbf{X};\mathbf{\Theta})$, $\forall\ i,k$, and their means and variances, as given by the formulae in (7) and (20), are used to compute the posterior expectations $\mathbb{E}[\cdot|\mathbf{Y},\mathbf{X}]$ in (15) and (16).

After (15)-(16) are computed and the TGGM parameters in $\mathbf{\Theta}$ are improved based on the gradient ascent in (13) , one iteration of the ML estimator is completed. Given the updated $\mathbf{\Theta}$, one then repeat the cycles with (22) to find the approximate posteriors and again make another update of $\mathbf{\Theta}$, and so on. The complete ML estimation algorithm is

summarized in Algorithm 1, where T_1 represents the number of cycles with (22) to find the best posterior distribution $q(\mathbf{H}|\widetilde{\mathbf{\Theta}})$ for each newly-updated $\mathbf{\Theta}$. We can see that the complexity mainly comes from the estimation of expectation $\mathbb{E}[\mathbf{h}_i|\mathbf{y}_i, \mathbf{x}_i]$, which is $\mathcal{O}(T_1 M^2)$.

Algorithm 1 ML Estimator for TGGM Regression

Randomly initialize the model parameters $\mathbf{\Theta}$;
repeat
  for $t = 1$ **to** T_1 **do**
    for $k = 1$ **to** M **do**
      Update $\mathbb{E}[\mathbf{h}_i(k)|\mathbf{y}_i, \mathbf{x}_i]$ using (7);
      Replace the k-th value of $\mathbb{E}[\mathbf{h}_i|\mathbf{y}_i, \mathbf{x}_i]$ with $\mathbb{E}[\mathbf{h}_i(k)|\mathbf{y}_i, \mathbf{x}_i]$;
    end for
  end for
  Compute $\mathbb{E}[\mathbf{h}_i\mathbf{h}_i^T|\mathbf{y}_i, \mathbf{x}_i]$ using (7) and (20);
  Calculate the gradients of log-likelihood using (15) and (16);
  Update model parameters $\mathbf{\Theta}$ with gradient ascend;
until Convergence of log-likelihood

Finally, it should be noted that the expectations require frequent calculation of the ratio of $\frac{\phi(a)}{\Phi(a)}$. In practice, if it is computed directly, we easily encounter two issues. First, repeated computation of the integration $\Phi(a) = \int_{-\infty}^{a} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz$ is a waste of time; second, when a is small, e.g. $a \leq -37$, the CDF $\Phi(a)$ and the PDF $\phi(a)$ are so tiny that a double-precision floating number can no longer represent them accurately. If we compute them with the double-precision numbers, we easily encounter the issue of $\frac{0}{0}$. Fortunately, both these issues can be solved by using a lookup table. Specifically, we pre-compute $\frac{\phi(a)}{\Phi(a)}$ at densely-sampled discrete values of a using high-accuracy computation, such as the symbolic calculation in Matlab, and store the results in a table. When we need $\frac{\phi(b)}{\Phi(b)}$ for any b, we look for two values of a that are closest to b and use the interpolation of the two $\frac{\phi(a)}{\Phi(a)}$ to estimate $\frac{\phi(b)}{\Phi(b)}$.

4. Extension to Other Learning Tasks

4.1. Nonlinear Classification

Let $c \in \{1, \cdots, n\}$ denote n possible classes. Let $\mathbf{T}_c \in \mathbb{R}^{(n-1)\times n}$ be a class-dependent matrix obtained from $-\mathbf{I}_n$ by setting the c-th column to one and deleting the c-th row (Liao et al., 2007). We define a nonlinear classifier as

$$p(c) = \int_0^\infty \int_0^\infty \mathcal{N}\left(\mathbf{z}\,|\mathbf{T}_c(\mathbf{W}_1\mathbf{h} + \mathbf{b}_1), \mathbf{T}_c\mathbf{T}_c^T\right) d\mathbf{z} \times \mathcal{N}_T\left(\mathbf{h}\,|\mathbf{W}_0\mathbf{x} + \mathbf{b}_0, \sigma_0^2\mathbf{I}_m\right) d\mathbf{h}. \quad (23)$$

The inner integral is due to the multinomial probit model (Albert & Chib, 1993) which transforms the TGGM's output vector $\mathbf{y}$ in (1) into a class label according to $c = \arg\max_k \mathbf{y}(k) = \arg\max_k \mathbb{I}(\mathbf{T}_k\mathbf{y} \geq \mathbf{0})$. Therefore, $p(c) = p(\mathbf{T}_c\mathbf{y} \geq \mathbf{0}) = \int_{\mathbf{T}_c\mathbf{y}\geq\mathbf{0}} \mathcal{N}(\mathbf{y}\,|\mathbf{W}_1\mathbf{h} + \mathbf{b}_1, \mathbf{I}_n)\, d\mathbf{y}$. A change of variables $\mathbf{z} \triangleq \mathbf{T}_c\mathbf{y}$ leads to $p(c) = \int_0^\infty \mathcal{N}\left(\mathbf{z}\,|\mathbf{T}_c(\mathbf{W}_1\mathbf{h} + \mathbf{b}_1), \mathbf{T}_c\mathbf{T}_c^T\right) d\mathbf{z}$.

The model described by (23) can be trained by an ML estimator, similarly to the case of regression, with the main difference being the additional latent vector $\mathbf{z}$, which can be treated in a similar way as $\mathbf{h}$. The posterior $p(\mathbf{z}, \mathbf{h}|\mathbf{x}, c)$ is still a truncated Gaussian distribution, whose moments can be computed using the methods in Section 3. The model predicts the class label of $\mathbf{x}$ using the rule $\hat{c} = \arg\max_k \mathbb{E}[\mathbf{y}(k)|\mathbf{x}]$, where $\mathbb{E}[\mathbf{y}|\mathbf{x}] = \mathbf{W}_1\mathbb{E}[\mathbf{h}|\mathbf{x}] + \mathbf{b}_1$ and $\mathbb{E}[\mathbf{h}(k)|\mathbf{x}_i] = g\left(\mathbf{W}_0(k,:)\mathbf{x} + \mathbf{b}_0(k), \sigma_0\right)$.

4.2. Deep Learning

The TGGM defined in (1) can be viewed as a neural network, where the input, hidden, and output layers are respectively constituted by $\mathbf{y}$, $\mathbf{h}$, and $\mathbf{x}$, and the hidden layer has outgoing connections to the output layer and incoming connections from the input layer. The topology of the hidden layer is determined by $\mathbf{P}_0$. So far, we have focused on $\mathbf{P}_0 = \sigma_0^2\mathbf{I}_m$, which defines a single layer of hidden nodes that are not interconnected. By using a more sophisticated $\mathbf{P}_0$, we can construct a deep TGGM with two or more hidden layers and enhanced representational versatility.

As an example, we let $\mathbf{h} \triangleq [\mathbf{h}^{(1)}; \mathbf{h}^{(2)}]$ and define $p(\mathbf{h}|\mathbf{x}) \propto \exp\{-\frac{1}{2\sigma_0^{(1)2}}\|\mathbf{h}^{(1)} - \mathbf{W}_0^{(1)}\mathbf{x} - \mathbf{b}_0^{(1)}\|^2\} \times \exp\{-\frac{1}{2\sigma_0^{(2)2}}\|\mathbf{h}^{(2)} - \mathbf{W}_0^{(2)}\mathbf{h}^{(1)} - \mathbf{b}_0^{(2)}\|^2\}\mathbb{I}(\mathbf{h} \geq \mathbf{0})$. Taking into account the normalization, the distribution can be written as $p(\mathbf{h}) = \mathcal{N}_T\left(\mathbf{h}\,|\boldsymbol{\zeta}, \mathbf{P}_0^{-1}\right)$, where $\boldsymbol{\zeta}$ and $\mathbf{P}_0$ depend on $\{\mathbf{W}_0^{(t)}, \mathbf{b}_0^{(t)}, \sigma_0^{(t)2}\}_{t=1}^2$. This distribution, along with $p(\mathbf{y}|\mathbf{h}^{(2)}) = \mathcal{N}(\mathbf{y}|\mathbf{W}_1\mathbf{h}^{(2)} + \mathbf{b}_1, \sigma_1^2\mathbf{I}_n)$, yields a TGGM with two hidden layers. Extensions to three or more hidden layers and to classification can be constructed similarly. A deep TGGM can be learned by using EM to maximize the likelihood, wherein the derivatives of the lower bound $\mathcal{Q}$ can be represented as $\frac{\partial\mathcal{Q}}{\partial\mathbf{\Theta}} = -\mathbb{E}\left[\frac{\partial E}{\partial\mathbf{\Theta}}|\mathbf{Y}, \mathbf{X}\right] + \mathbb{E}\left[\frac{\partial E}{\partial\mathbf{\Theta}}|\mathbf{X}\right]$, as in Section 3.1.

Training a multi-hidden-layer TGGM is almost the same as training a single-hidden-layer TGGM, except for the difference in estimating the prior expectation $\mathbb{E}[\frac{\partial E}{\partial\mathbf{\Theta}}|\mathbf{X}]$. In the single-hidden-layer case, since $\mathbf{P}_0$ is diagonal, $p(\mathbf{h}_i|\mathbf{x}_i; \mathbf{\Theta})$ factorizes into a set of univariate truncated normals, and therefore the expectation can be computed efficiently. In the multi-hidden-layer case, however, $p(\mathbf{h}_i|\mathbf{x}_i; \mathbf{\Theta})$ is a multivariate truncated normal, and thus the prior expectation is as difficult to compute as the posterior expectation. Following Section 3.3, we use mean-filed VB to approximate $p(\mathbf{h}_i|\mathbf{x}_i, \mathbf{\Theta})$ by factorized univariate truncated normals and

estimate $\mathbb{E}[\frac{\partial E}{\partial \mathbf{\Theta}}|\mathbf{X}]$ with the univariate distributions.

In practice, we find that starting from the VB approximation of $p(\mathbf{h}_i|\mathbf{y}_i, \mathbf{x}_i, \mathbf{\Theta})$ can improve the VB approximation of $p(\mathbf{h}_i|\mathbf{x}_i, \mathbf{\Theta})$, a feature similar to that observed in the contrastive divergence (CD) (Hinton, 2002).

5. Experiments

We report the performance of the proposed TGGM models on publicly available data sets, in comparison to competing models. In all experiments below, RMSProp (Tieleman & Hinton, 2012) is applied to update the model parameters by using the current estimated gradients, with RMSprop delay set to be 0.95.

Regression The root mean square error (RMSE), averaged over multiple trials of splitting each data set into training and testing subsets, is used as a performance measure to evaluate the TGGM against the ReLU neural network. The results reported in (Hernández-Lobato & Adams, 2015) are used as the reference to the performances of ReLU neural networks. The comparison is based on the same data and same training/testing protocols in (Hernández-Lobato & Adams, 2015), by using a consistent setting for the TGGM as follows: a single hidden layer is used in the TGGM for all data sets, with 100 hidden nodes used for Protein Structure and Year Prediction MSD, the two largest data sets, and 50 hidden nodes used for the other data sets.

Two methods, BP and ML estimation, are applied to train each TGGM, resulting in two versions of the TGGM for each data set, referred to TGGM-BP and TGGM-ML, respectively. For both training methods, $\mathbf{\Theta}$ is initialized as Gaussian random numbers, with each component a random draw from $\mathcal{N}(0, 0.01)$. To speed up, each gradient update uses a mini-batch of training samples, resulting in stochastic gradient search. The batch size is 100 for the two largest data sets and 50 for the others. For ML estimation, the number of cycles used by mean-field VB is set to 10, and $\sigma_1^2 = \sigma_0^2 = 0.5$.

The testing RMSE's of the TGGM are summarized in Table 1, alongside the corresponding results (Hernández-Lobato & Adams, 2015) for the ReLU neural networks trained by BP and probabilistic backpropagation (PBP). BP provides a point estimate of the model parameters while PBP provides the posterior distribution. It is seen from Table 1 that TGGM-BP performs slightly better than ReLU on most data sets. The gain may be attributed to the soft activation function $g(\mu, \sigma)$ which provides the freedom in choosing appropriate σ_0 according to data's characteristics, unlike ReLU which fixes σ to 0. The nonzero slope in $g(\mu, \sigma)$ for $\mu < 0$ may be another contributing factor, as it has been shown in (He et al., 2015) that replacing the zero part of ReLU with a sloping line leads to better results.

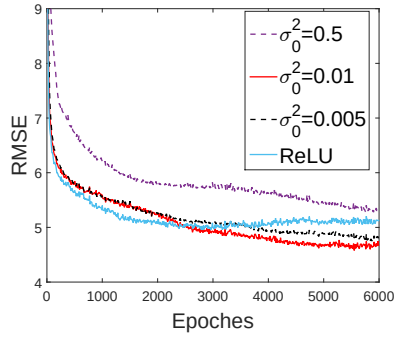

Figure 2. Illustration of the impact of σ_0^2 on TGGM-BP, based on the RMSEs for Concrete Strength in a single trial.

Furthermore, it can be observed that TGGM-ML outperforms TGGM-BP on most data sets. This is because ML-based training fully exploits the flexibilities provided by a probabilistic model and, as discussed in Section 3.2, is more accurate in reflecting the underlying model, in contrast to TGGM-BP which made a wrong assumption about the model at the very beginning.

We also observe that, if σ_0^2 is set close to 0, the performance of TGGM-BP approaches that of the ReLU neural network. This is not surprising since $g(\mu, \sigma_0)$ approaches the ReLU activation function as $\sigma_0^2 \to 0$. As we increase the value of σ_0^2, the TGGM's performance improves gradually, until it reaches a saturating value. This is reasonable because $g(\mu, \sigma_0)$ becomes more linearly (w.r.t. μ) as σ_0^2 becomes larger, which weakens its nonlinear representational abilities. Empirically, we find that TGGM-BP performs similarly within an appropriate range of σ_0^2. The results in Table 1 are based on $\sigma_0^2 = 0.01$, which is found to be a good setting for all data sets. The impact of σ_0^2 on the RMSE results is illustrated in Fig. 2. Note that σ_0^2 can also be learned directly from the data, which is an interesting future work.

We found that the optimal σ_0^2 for TGGM-ML is typically larger than that for TGGM-BP. This is perhaps because a larger σ_0^2 provides increased flexibility in the TGGM, which can be exploited by a probabilistic inference method like expectation-maximization. As a result, we use $\sigma_0^2 = 0.5$ for TGGM-ML in the experiments.

Classification Three public benchmark data sets are considered for this task: MNIST, 20 NewsGroups, and Blog. The MNIST data set includes 60,000 training images and 10,000 testing images of handwritten digits of zero through ten. The 20 NewsGroups data sets is composed of 18,845 documents, written with a vocabulary of 2,000 words, from 20 different groups, with the data partitioned into a training set of 11,315 documents and a testing set of 7,531 documents (Li et al., 2016). The Blog data set contains 13,245 documents, written with 17,292 words, about the US presidential elections; the data are partitioned into 7,832 train-

Table 1. Averaged Test RMSE and Std. Errors

Dataset	N	d	ReLU-BP	ReLU-PBP	TGGM-BP	TGGM-ML
Boston Housing	506	13	3.228±0.1951	3.014± 0.1800	2.927 ± 0.2910	**2.820± 0.2565**
Concrete Strength	1030	8	5.977± 0.0933	5.667± 0.0933	5.657 ± 0.2685	**5.395± 0.2404**
Energy Efficiency	768	8	1.098 ± 0.0738	1.804 ± 0.0481	**1.029 ± 0.1206**	1.244 ± 0.0979
Kin8nm	8192	8	0.091± 0.0015	0.098± 0.0007	0.088 ± 0.0025	**0.083 ± 0.0034**
Naval Propulsion	11934	16	0.001± 0.0001	0.006± 0.0000	**0.00057± 0.0001**	0.003 ± 0.0002
Cycle Power Plant	9568	4	4.182± 0.0402	4.124± 0.0345	**3.949 ± 0.1478**	4.183 ± 0.0955
Protein Structure	45730	9	4.539± 0.0288	4.732± 0.0130	4.477± 0.0483	**4.431 ± 0.0292**
Wine Quality Red	1599	11	0.645± 0.0098	0.635±0.0079	0.640 ± 0.0469	**0.625 ± 0.0340**
Yacht Hydrodynamic	308	6	1.182± 0.1645	1.015± 0.0542	0.957 ± 0.2319	**0.841 ± 0.2028**
Year Prediction MSD	515,345	90	8.932 ± N/A	**8.878 ± N/A**	8.918 ± N/A	9.002 ± N/A

ing documents and 5,413 testing documents (Chen et al., 2015). One-hidden-layer and two-hidden-layer TGGM models are considered, with each hidden layer containing 100 or 200 nodes. Similar to the regression model, the TGGM classifier is trained by both BP and ML, with the resulting models termed as TGGM-BP and TGGM-ML, respectively.

The models are randomly initialized with Gaussian random numbers drawn from $\mathcal{N}(0, 0.01)$. The step-size for gradient ascent is chosen from $[10^{-4}, 5 \times 10^{-3}]$ by maximizing the accuracy on a cross-validation set. The TGGMs use a minibatch size of 500 for MNIST and 200 for the other two data sets, while the ReLU uses 100 for all data sets. Variance parameters $\{\sigma_0^2, \sigma_1^2\}$ are set to 0.5 for TGGM-ML and 0.01 for TGGM-BP, in both one and two-layer models. When ML estimation is applied, the number of VB cycles is initially set to 30 and then gradually decreases to 5. The data sets are also used to train and test a ReLU neural network implemented in Caffe (Jia et al., 2014), to produce the competing results for comparison. From Table 2, it can be seen that TGGM-BP generally outperforms ReLU on the two document data sets and maintains a comparable performance on the image data set, for both one- and two-hidden-layer models. It is further observed that TGGM-ML has the best performance on all three data sets, with the best performance achieved by the two-hidden-layer models on MNIST and Blog, and by the one-hidden-layer model on 20 NewsGroup.

Table 2. Test Accuracy of Classification

Methods	MNIST	20 News	Blog
ReLU (100)	97.58%	72.8%	65.86%
ReLU (200)	97.89%	73.27%	67.02%
ReLU (100-100)	97.83%	69.94%	67.93%
ReLU (200-200)	98.04%	69.91%	65.07%
TGGM-BP (100)	97.52%	73.65%	67.50%
TGGM-BP (200)	97.56%	73.62%	67.52%
TGGM-BP (100-100)	97.76%	71.06%	66.82%
TGGM-BP (200-200)	98.12%	71.18%	67.73%
TGGM-ML (100)	97.75%	**73.74%**	69.83%
TGGM-ML (200)	97.97%	73.38%	69.75%
TGGM-ML (100-100)	98.05%	68.01%	**69.89%**
TGGM-ML(200-200)	**98.31%**	67.52%	66.64%

6. Conclusions

We have proposed a nonlinear statistical learning framework termed TGGM. By introducing truncated latent variables into the traditional GGM, we obtain the TGGM as a non-Gaussian nonlinear model with significantly enhanced modeling ability compared to the GGM. We demonstrate that regression and classification can be realized through appropriately constructed TGGMs. With carefully designed graphical structures, deep versions of TGGMs have also been obtained. It is shown that, for regression and classification, TGGMs can be approximately viewed as a deterministic neural network with an activation function similar to ReLU. Because of this, TGGMs can be trained with BP. However, BP does not exactly maximize the likelihood of a TGGM, due to the inherent Gaussian assumption it makes. To overcome this limitation, we have developed an algorithm to correctly maximize the likelihood under the truncated Gaussian assumption. Experimental results show that the TGGM trained by BP generally performs better than the ReLU network, indicating the advantage of the new activation function. It is further shown that the TGGM trained by ML learning achieves the best performance on most data sets in consideration. It should be emphasized that the tasks considered in this paper are only specific applications of the TGGM framework under special forms of the precision matrices. In the future, we will consider TGGMs with lateral connections between hidden nodes. We may also generalize the supervised TGGM to the unsupervised case, using constructs similar to RBMs. Moreover, investigation of how the quality of uncertainty estimates affects the performance is also of interest.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable and constructive comments. This research was supported in part by ARO, DARPA, DOE, NGA and ONR.

References

Albert, James H and Chib, Siddhartha. Bayesian analysis of binary and polychotomous response data. *Journal of the American statistical Association*, 88(422):669–679, 1993.

Chen, Changyou, Buntine, Wray, Ding, Ni, Xie, Lihua, and Du, Liang. Differential topic models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(2): 230–242, 2015.

Corduneanu, A. and Bishop, C. Variational Bayesian model selection for mixture distributions. In *AI and Statistics*, pp. 27–34, 2001.

Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:1–38, 1977.

Downs, Oliver B, MacKay, David JC, Lee, Daniel D, et al. The nonnegative boltzmann machine. In *NIPS*, pp. 428–434, 1999.

Frey, Brendan J. Continuous sigmoidal belief networks trained using slice sampling. *Advances in Neural Information Processing Systems*, pp. 452–458, 1997.

Frey, Brendan J and Hinton, Geoffrey E. Variational learning in nonlinear gaussian belief networks. *Neural Computation*, 11(1):193–213, 1999.

Galbraith, JI, Moustaki, Irini, Bartholomew, David J, and Steele, Fiona. *The analysis and interpretation of multivariate data for social scientists*. CRC Press, 2002.

Gan, Zhe, Henao, Ricardo, Carlson, David E, and Carin, Lawrence. Learning deep sigmoid belief networks with data augmentation. In *AISTATS*, 2015.

Genz, Alan. Numerical computation of multivariate normal probabilities. *Journal of computational and graphical statistics*, 1(2):141–149, 1992.

Genz, Alan and Bretz, Frank. *Computation of multivariate normal and t probabilities*, volume 195. Springer Science & Business Media, 2009.

Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

Hernández-Lobato, José Miguel and Adams, Ryan P. Probabilistic backpropagation for scalable learning of bayesian neural networks. *Proceedings of The 32nd International Conference on Machine Learning*, 2015.

Hinton, G. E. and Ghahramani, Z. Generative models for discovering sparse distributed representations. *Phil. Trans. Roy. Soc., B*, 352:1177–90, 1997.

Hinton, Geoffrey E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14 (8):1771–1800, 2002.

Hinton, Geoffrey E, Osindero, Simon, and Teh, Yee-Whye. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Honorio, Jean, Samaras, Dimitris, Paragios, Nikos, Goldstein, Rita, and Ortiz, Luis E. Sparse and locally constant gaussian graphical models. In *Advances in Neural Information Processing Systems*, pp. 745–753, 2009.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Johnson, Norman L, Kotz, Samuel, and Balakrishnan, Narayanaswamy. Continuous univariate distributions, vol. 1-2, 1994.

Jordan, Michael I, Ghahramani, Zoubin, Jaakkola, Tommi S, and Saul, Lawrence K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Koller, Daphne and Friedman, Nir. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Li, Chunyuan, Stevens, Andrew, Chen, Changyou, Pu, Yunchen, Gan, Zhen, and Carin, Lawrence. Learning weight uncertainty with stochastic gradient mcmc for shape classification. In *CVPR*, 2016.

Liao, Xuejun, Li, Hui, and Carin, Lawrence. Quadratically gated mixture of experts for incomplete data classification. In *Proceedings of the 24th International Conference on Machine learning*, pp. 553–560. ACM, 2007.

Liu, Ying and Willsky, Alan. Learning gaussian graphical models with observed or latent fvss. In *Advances in*

Neural Information Processing Systems, pp. 1833–1841, 2013.

Meng, Zhaoshi, Eriksson, Brian, and Hero, Al. Learning latent variable gaussian graphical models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1269–1277, 2014.

Mudholkar, Govind S and Hutson, Alan D. The epsilon–skew–normal distribution for analyzing near-normal data. *Journal of Statistical Planning and Inference*, 83 (2):291–309, 2000.

Neal, Radford M. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113, 1992.

Oh, Jung Hun and Deasy, Joseph O. Inference of radio-responsive gene regulatory networks using the graphical lasso algorithm. *BMC bioinformatics*, 15(Suppl 7):S5, 2014.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

Salakhutdinov, Ruslan and Hinton, Geoffrey E. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pp. 448–455, 2009.

Socci, Nicholas D, Lee, Daniel D, and Sebastian Seung, H. The rectified gaussian distribution. *Advances in Neural Information Processing Systems*, pp. 350–356, 1998.

Su, Qinliang and Wu, Yik-Chung. On convergence conditions of gaussian belief propagation. *Signal Processing, IEEE Transactions on*, 63(5):1144–1155, 2015a.

Su, Qinliang and Wu, Yik-Chung. Distributed estimation of variance in gaussian graphical model via belief propagation: Accuracy analysis and improvement. *Signal Processing, IEEE Transactions on*, 63(23):6258–6271, 2015b.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.