Interpretable Latent Variable Models for Graphs

UGP Presentation

Gurpreet Singh
24th November, 2018

Indian Institute of Technology, Kanpur

Background

BBVI - Overview

In Variational Inference, the optimization objective is given by the ELBO

$$\log\, q\left(\mathbf{x}\right) \geq \mathcal{L}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \mathop{\mathbb{E}}_{\mathbf{z} \sim q(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi})} \left[\log\, p(\mathbf{x}, \mathbf{z}) - \log\, q(\mathbf{z} \,\middle|\, \mathbf{x}, \boldsymbol{\phi}) \right]$$

Black Box Variational Inference is based on stochastic optimization of the variational objective

$$\nabla \mathcal{L}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \mathop{\mathbb{E}}_{q(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi})} \left[\nabla_{\boldsymbol{\phi}} \log\, q(\mathbf{z} \,\middle|\, \mathbf{x}, \phi) \left(\log\, p(\mathbf{x}, \mathbf{z}) - \log\, q(\mathbf{z} \,\middle|\, \mathbf{x}, \boldsymbol{\phi}) \right) \right]$$

The above term is known as the Score Function.

BBVI - Reparameterization

If we can find a reparametrization $\boldsymbol{\epsilon} = f^{-1}(\mathbf{z}, \boldsymbol{\phi})$ such that $\boldsymbol{\epsilon}$ is independent of $\boldsymbol{\phi}$, then

$$\nabla \mathcal{L}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \mathop{\mathbb{E}}_{q(\boldsymbol{\epsilon})}\left[\nabla_{\mathbf{z}} \left[\log\, p(\mathbf{x}, \mathbf{z}) - \log\, q(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi}) \right]_{\mathbf{z}=f(\boldsymbol{\epsilon}, \boldsymbol{\phi})} \nabla_{\boldsymbol{\phi}} f(\boldsymbol{\epsilon}, \boldsymbol{\phi}) \right]$$

If $\mathbf{z}$ is discrete $p(\mathbf{z})$ is a p.m.f. and therefore, the derivatives no longer exist. Hence, reparameterization is **not** possible

Discrete Latent Variables

Some methods to train discrete latent variable models -

1. Marginalize all discrete latent variables
2. Using local expectation gradients, and reparametrization and marginalization
3. Relaxed computation of discrete densities (straight through approach)
4. Smooth relaxations for discrete variables (e.g. Concrete Distribution)
5. Using the Score Function approach with variance reduction using control variates/non-uniform sampling.

Gumbel-Softmax Trick

The Concrete Distribution

Sampling from a categorical distribution: Sample U from a $[\,0,1\,]$ uniform distribution, then reparameterize as

$$\mathrm{C} = \operatorname*{arg\,max}_{k \in [\,K\,]} \log \alpha_k - \log\left(-\log(\mathrm{U}_k)\right)$$

A softmax version of this can be used as smooth relaxations

$$\mathrm{Z}_k = \frac{\exp\Big(\left(\log \alpha_k + G_k\right)/\tau\Big)}{\sum_{k'=1}^{K} \exp\Big(\left(\log \alpha_{k'} + G_{k'}\right)/\tau\Big)}$$

where τ is the **temperature** parameter.

Discrete Variational Autoencoders

Discrete Variational Autoencoders

Gumbel-Softmax trick only works for factorial priors. Discrete Variational Autoencoders are models which work with more expressive priors, the (Restricted) Boltzmann Machine priors.

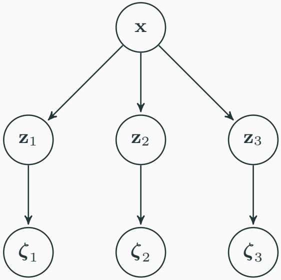
Boltzmann Machine Prior

$$q(\mathbf{z}) = \frac{e^{-E(\mathbf{z})}}{Z_\theta}, \qquad \text{where}$$
$$E(\mathbf{z}) = \mathbf{z}^{\mathrm{T}}\mathbf{b} + \mathbf{z}^{\mathrm{T}}\mathrm{W}\mathbf{z}$$

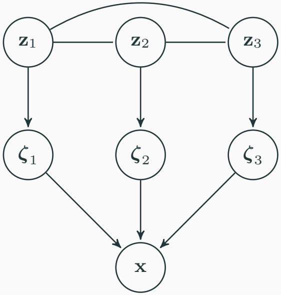
Restricted Boltzmann Machine Prior

$$E(\mathbf{z}) = {\mathbf{z}_1}^{\mathrm{T}}\mathbf{b}_1 + {\mathbf{z}_2}^{\mathrm{T}}\mathbf{b}_2 + {\mathbf{z}_1}^{\mathrm{T}}\mathrm{W}\mathbf{z}_2$$

Discrete Variational Autoencoders



(a) Posterior Network ($q\left(\boldsymbol{\zeta}, \mathbf{z} \mid \mathbf{x}\right)$)



(b) Prior Network ($p\left(\boldsymbol{\zeta}, \mathbf{z}\right)$)

Figure 1: Graphical Models of the approximating posterior (a) and the prior (b)

Discrete Variational Autoencoders

The prior $p(\boldsymbol{\zeta}, \mathbf{z} \,|\, \boldsymbol{\theta})$ is suitably written as

$$\begin{aligned} p\left(\boldsymbol{\zeta}, \mathbf{z} \,\middle|\, \boldsymbol{\theta}\right) &= r\left(\boldsymbol{\zeta} \,\middle|\, \mathbf{z}\right) \cdot p\left(\mathbf{z} \,\middle|\, \boldsymbol{\theta}\right), \qquad \text{where} \\ r\left(\boldsymbol{\zeta} \,\middle|\, \mathbf{z}\right) &= \prod_i r\left(\zeta_i \,\middle|\, z_i\right) \end{aligned}$$

The posterior is correspondingly augmented as

$$q\left(\boldsymbol{\zeta}, \mathbf{z} \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right) = r\left(\boldsymbol{\zeta} \,\middle|\, \mathbf{z}\right) \cdot q\left(\mathbf{z} \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right)$$

The log-likelihood is assumed to have the following property

$$q\left(\mathbf{x} \,\middle|\, \mathbf{z}, \boldsymbol{\zeta}, \boldsymbol{\theta}\right) = q\left(\mathbf{x} \,\middle|\, \boldsymbol{\zeta}, \boldsymbol{\theta}\right)$$

DVAEs - Computing the Gradients

In the VAE framework, we decomompose the ELBO as

$$\mathcal{L}\left(\mathbf{z}, \boldsymbol{\phi}\right) = \underbrace{-\mathrm{KL}\left(q\left(\boldsymbol{\zeta}, \mathbf{z} \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right) \,||\, p\left(\boldsymbol{\zeta}, \mathbf{z} \,\middle|\, \boldsymbol{\theta}\right)\right)}_{\text{KL Term}} + \underbrace{\mathop{\mathbb{E}}_{q(\boldsymbol{\zeta}, \mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi})}\left[p(\mathbf{x} \,|\, \boldsymbol{\zeta}, \boldsymbol{\theta})\right]}_{\text{Autoencoding Term}}$$

The gradients of the KL term are given as

$$\nabla \mathrm{KL}\left(q \,\middle|\middle|\, p\right) = \underbrace{\mathop{\mathbb{E}}_{q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi})}\left[\nabla E_{\boldsymbol{\theta}}(\mathbf{z})\right] - \mathop{\mathbb{E}}_{p(\mathbf{z} \mid \boldsymbol{\theta})}\left[\nabla E_{\boldsymbol{\theta}}(\mathbf{z})\right]}_{\text{Gradients for Prior Parameters}} +$$

$$\underbrace{\sum_i \sum_{z_i} q\left(z_i \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right) \log\, q\left(z_i \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right) + \nabla E_{\boldsymbol{\theta}}\left(\mathop{\mathbb{E}}_{q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi})}\left[\mathbf{z}\right]\right)}_{\text{Gradients for Posterior Parameters}}$$

DVAE - Spike-and-Exponential Smoothing

$r\left(\zeta_i \,\middle|\, z_i\right)$ for spike-and-exp transformation is given as

$$r\left(\zeta_i \,\middle|\, z_i = 0\right) = \delta_0(\zeta_i)$$
$$r\left(\zeta_i \,\middle|\, z_i = 1\right) = \frac{\beta e^{\beta\zeta}}{e^{\beta} - 1}, \text{ if } \zeta_i \in [\,0, 1\,] \text{ else, } 0$$

The inverse CDF for the same is given as

$$\zeta_i = F^{-1}_{q(\zeta_i \,|\, \mathbf{x}, \boldsymbol{\phi})}(\rho_i) = \max\left\{\frac{1}{\beta} \cdot \log\left[\left(\frac{\rho_i + q_i - 1}{q_i}\right) \cdot \left(e^{\beta} - 1\right)\right] + 1, 0\right\}$$

where $\rho_i \in (0, 1)$ is sampled from a uniform distribution (from 0 to 1), and q_i is the short-hand notation for $q\left(z_i \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right)$.

DVAE++ - Exp-and-Exp smoothing

$r\left(\zeta_i \,\middle|\, z_i\right)$ for exp-and-exp transformation, similar to the spike-and-exp case, is given as

$$r\left(\zeta_i \,\middle|\, z_i = 0\right) = \frac{\beta e^{\beta(1-\zeta)}}{e^{\beta} - 1}, \text{ if } \zeta_i \in [\,0, 1\,] \text{ else, } 0$$

$$r\left(\zeta_i \,\middle|\, z_i = 0\right) = \frac{\beta e^{\beta\zeta}}{e^{\beta} - 1}, \text{ if } \zeta_i \in [\,0, 1\,] \text{ else, } 0$$

The inverse CDF in this case is given (details in the original paper) as

$$\zeta_i = F^{-1}_{q(\zeta_i \,|\, \mathbf{x}, \boldsymbol{\phi})}(\rho_i) = -\frac{1}{\beta} \cdot \log \frac{-b + \sqrt{b^2 - 4ac}}{2}$$

where $\rho \in (0, 1)$ is sampled from a uniform distribution (from 0 to 1), and $b = \left[\rho + e^{-\beta}(q_i - \rho)\right] / (1 - q_i) - 1$, $c = \left(q_i e^{-\beta}\right) / (1 - q_i)$, and q_i is the short-hand notation for $q\left(z_i \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right)$.

The Gumbolt Trick

Gumbolt - Relaxed ELBO

The posterior and the prior for the smoothing variables are given as

$$p\left(\boldsymbol{\zeta} \,\middle|\, \boldsymbol{\theta}\right) = \frac{e^{-E_{\boldsymbol{\theta}}(\boldsymbol{\zeta})}}{\widetilde{Z}_{\boldsymbol{\theta}}}, \text{ where } \widetilde{Z}_{\boldsymbol{\theta}} = \int_{\boldsymbol{\zeta}} e^{E_{\boldsymbol{\theta}}(\boldsymbol{\zeta})} \, \mathrm{d}\boldsymbol{\zeta}$$
$$q\left(\boldsymbol{\zeta} \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right) = \prod_i \zeta_i q_i + (1 - \zeta_i)(1 - q_i)$$

The alternate ELBO, therefore, is given as

$$\widetilde{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \mathop{\mathbb{E}}_{q(\boldsymbol{\zeta} \mid \mathbf{x}, \boldsymbol{\phi})} \left[\log p\left(\mathbf{x} \,\middle|\, \boldsymbol{\zeta}, \boldsymbol{\theta}\right) - \log q\left(\boldsymbol{\zeta} \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right) - E_{\boldsymbol{\theta}}(\boldsymbol{\zeta}) \right] - \log \widetilde{Z}_{\boldsymbol{\theta}}$$

Gumbolt - Relaxed ELBO

Relaxed ELBO is given as

$$\breve{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \mathop{\mathbb{E}}_{q(\boldsymbol{\zeta} \mid \mathbf{x}, \boldsymbol{\phi})} \left[\log\, p\left(\mathbf{x} \,\middle|\, \boldsymbol{\zeta}, \boldsymbol{\theta}\right) - \log\, q\left(\boldsymbol{\zeta} \,\middle|\, \mathbf{x}, \boldsymbol{\phi}\right) - E_{\boldsymbol{\theta}}(\boldsymbol{\zeta}) \right] - \log\, Z_{\boldsymbol{\theta}}$$

where, we have

$$\log\, p\left(\mathbf{x} \,\middle|\, \boldsymbol{\theta}\right) \geq \widetilde{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right) \geq \breve{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right)$$

Also, as $\tau \to 0$, $\breve{\mathcal{L}} \to \widetilde{\mathcal{L}}$

Gumbolt - Regeneration

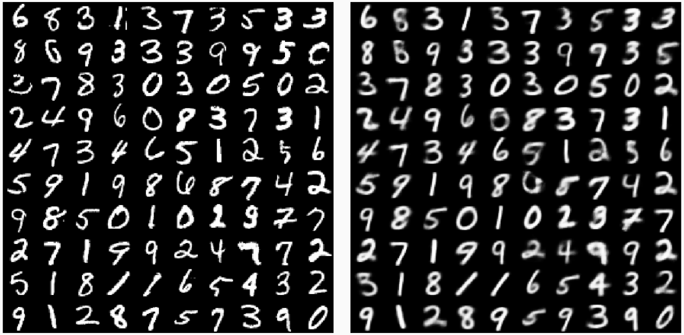


Figure 2: Regerenation Plot for Gumbolt-Vae

Gumbolt - Sampling

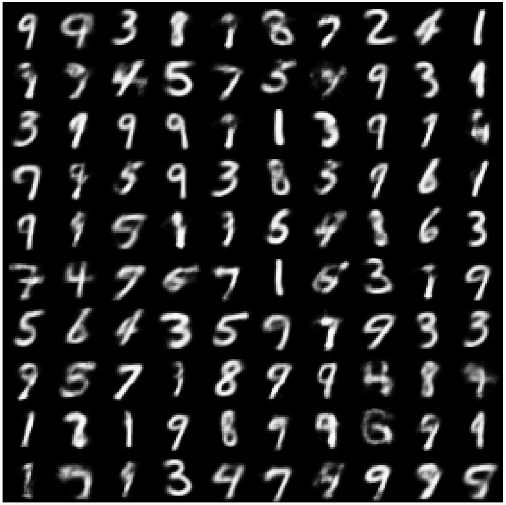


Figure 3: Sampling Plot for Gumbolt-Vae

Stochastic Block Models

Stochastic Block Models

A stochastic block model consists of

- A set of N nodes/vertices, possibly with features $\{\mathbf{x}_n\}$
- A partition of the nodes into K communities $\{\mathrm{C}_k\}$
- An adjacency matrix A of link probabilities between nodes

Trained using EM algorithm, alternating between community structure (latent variables) and link likelihood parameters.

Graph Variational Autoencoders

GCN: Like a convolutional neural network, the features of a node and its neighbours are convoluted to compute inputs for the next layer

GVAE: A GCN is used to compute parameters for the latent representations (encoder), and the latent representaions are passed into a standard neural network to regenerate link probabilities (decoder).

GVAEs use gaussian latent variables, and therefore the reparameterization is simple

Graph Variational Autoencoders

Encoder: Computes the means and log-variances of the latent variables using a GCN

$$\{\mu_{n,k}, \sigma_{n,k}\}_{k=1}^{K} = \mathrm{GCN}\left(\mathbf{x}_n, \mathrm{A}, \boldsymbol{\phi}\right)$$

Decoder: Generates edges/links (A) between nodes

$$\mathbf{z}_n \sim \mathcal{N}\left(\boldsymbol{\mu}_n, \mathrm{diag}(\boldsymbol{\sigma}_n^2)\right)$$
$$A_{n,m} \sim \sigma\left(\mathbf{f}(\mathbf{z}_n)^{\mathrm{T}}\mathbf{f}(\mathbf{z}_m)\right)$$

Affords fast inference and excellent predictive power, however latent representations are non-interpretable

Non-Parametric Latent Feature Relational Model

The latent features are binary vectors sampled using the Indian Buffet Process prior, and a weigh matrix sampled using a multivariate gaussian is used to compute link probabilities.

$$Z \sim \mathrm{IBP}(\alpha)$$
$$w_{k,k'} \sim \mathcal{N}\left(0, \sigma^2\right)$$
$$A_{n,m} \sim \sigma(\mathbf{z}_n{}^{\mathrm{T}} \mathrm{W} \mathbf{z}_m)$$

This produces interpretable features, however it uses MCMC algorithm and, therefore, suffers from slow inference.

Gumbolt-VGAE

Gumbolt Variational Graph Auto Encoder (Gumbolt-VGAE)

We try to combine the two approaches, *i.e.* have interpretable latent representations along with a powerful predictive model.

Binary latent vectors ($\mathbf{b}$) are used in company with real latent vectors ($\mathbf{r}$), and the Gumbolt model (with RBM prior) is employed to reparameterize the binary latent vectors. Final latent representation $\mathbf{z}$ is given as $\mathbf{z} = \mathbf{b} \odot \mathbf{r}$

Gumbolt-VGAE: VAE Encoder/Decoder

The Decoder

$$[\,\mathbf{b}_1, \mathbf{b}_2\,] \sim \mathrm{RBM}\left(\mathbf{a}_1, \mathbf{a}_2, \mathrm{W}\right), \qquad \mathbf{r} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

The Encoder

$$q\left(\mathbf{b}_n, \mathbf{r}_n \,\middle|\, \mathbf{x}, \mathrm{A}, \boldsymbol{\phi}\right) = \prod_{k=1}^{K} q\left(b_{n,k} \,\middle|\, \mathbf{x}_n, \mathrm{A}, \boldsymbol{\phi}\right) q\left(r_{n,k} \,\middle|\, \mathbf{x}_n, \mathrm{A}, \boldsymbol{\phi}\right)$$

and

$$q\left(b_{n,k} \,\middle|\, \mathbf{x}, \mathrm{A}, \boldsymbol{\phi}\right) = \mathrm{Bernoulli}\left(\pi_{n,k}\right)$$
$$q\left(r_{n,k} \,\middle|\, \mathbf{x}, \mathrm{A}, \boldsymbol{\phi}\right) = \mathcal{N}\left(\mu_{n,k}, \sigma^2_{n,k}\right)$$

where $\{\pi_{n,k}, \mu_{n,k}, \sigma_{n,k}\}_{k=1}^{K} = \mathrm{GCN}(\mathbf{x}_n, \mathrm{A})$.

Results

Method	Cora		Citeseer	
SC	0.8460	0.8850	0.8050	0.8500
DW	0.8310	0.8500	0.8050	0.8360
LFRM	0.9096	0.9060	0.8965	0.9118
VGAE	0.9260	0.9328	0.9200	0.9200
Gumbolt-VGAE	**0.9282**	**0.9396**	**0.9230**	**0.9339**

Table 1: Results (ROC and AP) of Gumbolt-VAE compared with baselines