Discrete Variational Autoencoders and Stochastic Block Models

Gurpreet Singh
150259

Abstract

Many latent variable models like variational graph autoencoders have been proposed recently for link prediction in graphs, however they provide little or no interpretability of the node representations. In contrast, variants of the Stochastic Block Models (SBMs) such as the mixed-membership model and non-parametric latent feature relational models have been successful in providing interpretable latent features along with adroitly discovering community structure. In an attempt to unify these two strands of research, this work presents an enhanced variational autoencoder model for graphs, with each node modelled as a binary latent vectors with an expressive restricted Boltzmann machine (RBM) prior. To handle the intractable inference, we leverage the Gumbolt reparametrization, along with continuous latent representations to retain the interpretability while still retaining excellent predictive power of these representations. We will also present a brief survey of some of the state of the art models for using binary latent variables in the variational inference literature, followed by some experimental results on the behaviour of these models.

1. Introduction

With the increasing importance of analyzing social networks, there has been a lot of work on link prediction for graphs or relational data.

Learning vector-valued node representations (a.k.a. embeddings) is a fundamental task in many problems involving graph-structured data. In a probabilistic modelling setting, we can learn these by treating the embeddings as latent variables and assuming that the graph adjacency matrix is generated via the interactions of these latent variables. Recent works, such as Kipf and Welling (2016) offer excellent predictive properties for this task, however, the caveat with using such models is the continuous nature of the latent representations, as these tend to not be interpretable.

Some models such as the Stochastic Block Model and its variants afford interpretable embeddings, but either suffer from low predictive power, or slow inference. In this work, these two approaches are combined to present a novel model which retains the predictive expressiveness of continuous latent variables, while simultaneously presenting interpretable latent variables.

2. Background

2.1. Black Box Variational Inference

Black Box Variational Inference (Ranganath et al., 2014) is based on stochastic optimization of the variational objective. This allows us to perform model free analysis, which standard variational inference does not afford. Black box inference estimates the gradients of the variational objective (the ELBO) using Monte Carlo samples from the proxy posterior distribution. This can be derived by computing the gradients of the ELBO, which is written as below.

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\phi}) = \mathop{\mathbb{E}}_{\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi})} \Big[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi}) \Big] \tag{1}$$

$$\implies \nabla \mathcal{L}(\mathbf{x}, \boldsymbol{\phi}) = \mathop{\mathbb{E}}_{\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi})} \Big[\nabla_{\boldsymbol{\phi}} \log q(\mathbf{z} \mid \mathbf{x}, \phi) \Big(\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi}) \Big) \Big] \tag{2}$$

The derivation of the above transformation can be found in the original paper by Ranganath et al. (2014). Since we formulated the gradient of the ELBO as an expectation, we can now use Monte Carlo estimation to get unbiased estimates of the ELBO gradient. This method is popular for the generality of the method, as it can be applied on most posterior estimation problems, with very few restrictions. However, the gradient estimates computed using this method have high variance, and therefore are often unsuitable to use in practical scenarios.

Many suggestions have been posed to overcome the variance problem. Most of them have common themes, and can be loosely classified as the following three classes

1. **Control Variates** Control variates are stochastic terms added to the monte carlo estimates so as to reduce the variance, while preventing any extra bias in the estimates. Johnson and Zhang (2013), Paisley et al. (2012), Wang et al. (2013) and Ranganath et al. (2014) present some techniques to reduce variance in BBVI gradient estimates using control variates.

2. **Non-uniform Sampling** The idea behind non-uniform sampling is based on selecting mini-batches which produce low variance estimates of the ELBO gradient. These samples are assigned higher probability of sampling, and therefore non-uniform sampling is employed. Works by Csiba and Richtárik (2016), Gopalan et al. (2012), Perekrestenko et al. (2017), etc. fall under this category.

3. **Reparametrization** The reparametrization trick allows estimation of the ELBO gradients using samples from a noise distribution with an apt (deterministic) transformation from the noise variable to the latent variable in question. Reparametrization trick reduces the variance by much, and is most desirable in many cases.

   This trick was successfully used by Kingma and Welling (2013) for amortized inference, by transorming the gaussian latent variables as a deterministic mapping from the random normal noise variable to the desired distribution. The reparametrization trick is of much interest, and I probe deeper into the many methods for reparametrization in Black Box Variation Inference, before moving towards studying black box inference of discrete latent variable models (with reparametrization).

   Another problem with reparametrization is that it disallows the use of discrete latent variables. This is discussed in more detail in later sections.

Another popular technique to reduce variance relies on Rao Blackwellization, also used by Ranganath et al. (2014) to reduce variance for the standard BBVI gradient estimates.

2.2. Reparametrization for BBVI

The formulation written in equation 2 is known as the Score Function method. In order to reparameterize, the latent variable is written as a deterministic transformation of a noise variable (say $\boldsymbol{\epsilon}$) using a function f such that $\mathbf{z} = f(\boldsymbol{\epsilon})$. Equation 2 can then be written alternately as

$$< ++ > \nabla \mathcal{L}\left(\mathbf{x}, \boldsymbol{\phi}\right) \quad = \quad \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}} \left[\nabla_{\boldsymbol{\phi}} \log q(f(\boldsymbol{\epsilon}) \,|\, \mathbf{x}, \boldsymbol{\phi}) \left(\log p(\mathbf{x}, f(\boldsymbol{\epsilon})) - \log q(f(\boldsymbol{\epsilon}) \,|\, \mathbf{x}, \boldsymbol{\phi}) \right) \right] < ++ > \quad (3)$$

The biggest problem with the reparametrization trick is the limited number of distributions to which the trick can be applied successfully. Moreover, this often includes model specific changes and therefore undermines the Block Box property of BBVI. There have been attempts to remedy this by allowing general reparametrization methods.

Ruiz et al. (2016) presents a generic reparameterization method for a large class of distributions by mapping the latent variables to random variables which are assumed to be only weakly dependent on the parameters of the posterior. Although this allows us to reparametrize a much greated class of models, however, this still requires finding random variables and commensurate invertible mapping for weak dependence.

Figurnov et al. (2018) further builds upon generalized reparameterization (Ruiz et al., 2016) by introducing implicit differentiation *i.e.* differentiation without explicitly inverting the mapping function between the latent and reparameterization variables. This allows for better convergence properties.

Another generic reparametrization technique is reparameterization through acceptance/rejection sampling (Naesseth et al., 2016). Since reparameterization only allows deterministic mapping from reparameterization variables to the latent variables, this technique augments reparameterization for variables sampled using Accept/Reject algorithms.

The common problem among all these methods is the incapability to handle discrete latent variables. In fact, it is not possible to use reparametrization with discrete latent variables. Therefore, score function methods need to be used, however, they have such a high variance they are practically unusable.

2.3. Discrete Latent Variables and BBVI

Although it is theoretically possible to learn models with discrete latent variables using the Score Function method, however, this naive approach gives estimates of the gradients which are impractically high variance. This variance can be reduced by adding control variates, which essentially translates to the REINFORCE strategy (a technique popular in the Reinforcement Learning framework).

Even with this reduced variance, we still face other difficulties. As Rolfe (2016) points out, to capture an efficient estimate of the variance of a D-dimensional latent variable, we need at least D samples. Since a model can effectively have 100s of latent variables, the number of samples required to estimate the gradients is huge, and therefore inefficient. On the contrary, when using reparametrization, even a single sample can effectively estimate the direction of the gradients, and therefore work effeciently in training such models.

Most methods to train models with discrete latent variables follow under one of five classes.

1. <++> Exhaustive approaches marginalize all discrete variables [25, 26] and which are not scalable to more than a few discrete variables.
2. Local expectation gradients [27] and reparameterization and marginalization [28] estimators compute low-variance estimates at the cost of multiple function evaluations per gradient. These

approaches can be applied to problems with a moderate number of latent variables.

3. Relaxed computation of discrete densities [29] replaces discrete variables with continuous relaxations for gradient computation. A variation of this approach, known as the straight-through technique, sets the gradient of binary variables to the gradient of their mean [30, 31].

4. Continuous relaxations of discrete distributions [32] replace discrete distributions with continuous ones and optimize a consistent objective. This method cannot be applied directly to Boltzmann distributions. The DVAE [19] solves this problem by pairing each binary variable with an auxiliary continuous variable. This approach is described in Sec. 2.

5. The REINFORCE estimator [33] (also known as the likelihood ratio [34] or score-function estimator) replaces the gradient of an expectation with the expectation of the gradient of the score function. This estimator has high variance, but many increasingly sophisticated methods provide lower variance estimators. NVIL [3] uses an input-dependent baseline, and MuProp [35] uses a first-order Taylor approximation along with an input-dependent baseline to reduce noise. VIMCO [36] trains an IWAE with binary latent variables and uses a leave-one-out scheme to define the baseline for each sample. REBAR [37] and its generalization RELAX [38] use the reparameterization of continuous distributions to define baselines. <++>

In the following sections, I discuss some methods to translate discrete latent variables to continuous variables, using smoothing techniques. For each relaxation, there lies a concomitant reparametrization method for fast inference and low variance estimates for the ELBO gradient. This discussion is mostly centered around binary latent variables.

2.4. Restricted Boltzmann Machines

A boltzmann machine is an undirected probabilistic energy based model, where the probability mass for a binary (vector) latent variable, $\mathbf{z}$, is given by

$$\mathbb{P}[\,\mathbf{z}\,] \quad = \quad \frac{\exp\left(-E_{\boldsymbol{\theta}}(\mathbf{z})\right)}{Z_{\boldsymbol{\theta}}} \tag{4}$$

where $E_{\boldsymbol{\theta}}(\mathbf{z})$ is the energy function, and $Z_{\boldsymbol{\theta}}$ is the partition function, given by

$$Z_{\boldsymbol{\theta}} \quad = \quad \sum_{\{\mathbf{z}\}} \exp\left(-E_{\boldsymbol{\theta}}(\mathbf{z})\right)$$

Since the dimension of the variable $\mathbf{z}$ can be large, it is impractical to compute the partition function, and therefore sampling needs to be used to estimate the value. In order to allow easier sampling (using Blocked Gibbs Sampling), we often assume the relation (undirected) between variables to be bipartite. This means that the set of variables $\mathbf{z}$ is divided into two parts, $(\mathbf{z}_1, \mathbf{z}_2)$, and it is assumed that there is no link among the variables in $\mathbf{z}_1$ and $\mathbf{z}_2$ separately. This effectively means that given $\mathbf{z}_1$, the probability of each variables in $\mathbf{z}_2$ is independent, and vice versa. Such a model is known as the Restricted Boltzmann Machine (RBM). In the RBM literature, the variables $\mathbf{z}$ are divided into visible and hidden variables, however since the models I discuss use RBMs as priors, we omit such distinction between the two sets of latent variables.

The energy function in case of RBMs is given as

$$E_{\boldsymbol{\theta}}(\mathbf{z}) \quad = \quad \mathbf{a}\cdot\mathbf{z}_1 + \mathbf{b}\cdot\mathbf{z}_2 + {\mathbf{z}_2}^{\mathrm{T}}\mathrm{W}\mathbf{z}_1 \tag{5}$$

where $\mathbf{a}$, $\mathbf{b}$ and W are the biases (on $\mathbf{z}_1$ and $\mathbf{z}_2$, respectively) and weights.

3. Gumbel-Softmax Trick

The Gumbel-Softmax Trick (Maddison et al., 2016; Jang et al., 2016) affords a smooth relaxation for categorical variables, based on the Gumbel distribution and the Sofmax trick (and hence the name).

Typically, sampling from a bernoulli involves firstly sampling from a uniform random variable (from 0 to 1), U and return the categorical variable C (with say, K classes) as follows

$$\mathrm{C} \quad = \quad \underset{k \in [\,K\,]}{\arg\min}\ \mathrm{U} - \sum_{k'=1}^{k} \pi_{k'}$$

Note. The parameters $\{\pi_k\}_{k=1}^K$ denote the probabilities of each class for the categorical distribution, and therefore $\sum_{k=1}^K \pi_k = 1$. Below, I alternately use weights $\{\alpha_k\}_{k=1}^K$ instead of probabilities, and therefore, we will have $\pi_k = \frac{\alpha_k}{\sum_{k'=1}^K \alpha_{k'}}$

An alternate way, however, to sample from the categorical distribution is by using the Gumbel-Max trick (upon which the Gumbel-Softmax is based). The Gumbel-Max trick says that in order to sample from a categorical distribution with weights $\{\alpha_k\}_{k=1}^K$, we sample K uniform random variables $\{\mathrm{U}_k\}_{k=1}^K$ and reparametrize as follows

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

The variable $-\log\left(-\log\left(U\right)\right)$ is known to be from the Gumbel disitribution. This alternate sampling strategy allows us to write the relaxed version of the random variable C (denoted by $\mathbf{Z} = [\,\mathrm{Z}_1 \ldots \mathrm{Z}_K\,]$). which is then given as

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

where G_k is a sample from the gumbel distribution, and τ is a parameter, known as the temperature. It is obvious to see that as $\tau \to 0$, the smooth relaxation $Z_k \to \mathcal{I}\ (C = k)$.

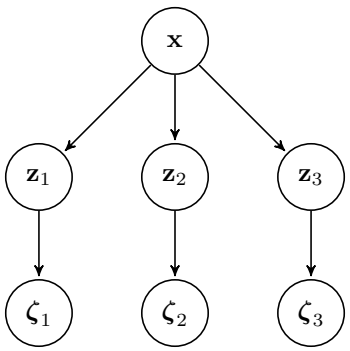
In the ELBO term, the terms for the prior and posterior of the latent variable C are replaced by simple relaxations to facilitate proper inference of the objective. This can be easily used to train binary latent variables, which is also shown in detail in Maddison et al. (2016). The problem, however, is that this relaxation is limited to simple categorical priors, and therefore does not scale to more complicated priors such as the Boltzman Machine prior.

In the following sections, some of the popular methods to infer models with Boltzman priors are discussed. These methods use alternate relaxations, which allow us to use better priors.

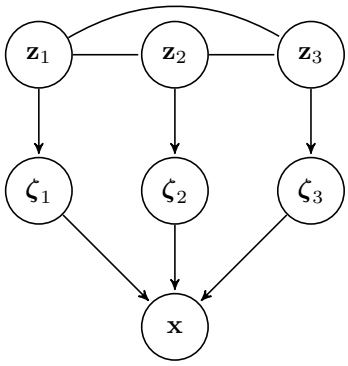
4. Discrete Variational Autoencoders

Rolfe (2016) proposed models with RBM priors within the variational autoencoder framework. Although Rolfe (2016) propose a model with autoregressive connections between the latent variables $\mathbf{z}$, thereby forming a hieararchy (in the posterior), we assume this to absent in our model. This simplifies the discussion and allows us to focus on the discussion pertaining to inference with binary latent variables.

In order to allow inference, the model is augmented with continuous variables $\boldsymbol{\zeta}$ as shown in Figure 1. Note, the model assumes that in the generative network, the observed labels $\mathbf{x}$ only depend on the smooth analogs of the latent variables, *i.e.* $\mathbf{x}$ only depend on $\boldsymbol{\zeta}$.



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



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

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

We can write the same mathematically, with the prior $p(\boldsymbol{\zeta}, \mathbf{z} \,|\, \boldsymbol{\theta})$ as follows

$$\begin{aligned} p\left(\boldsymbol{\zeta}, \mathbf{z} \,|\, \boldsymbol{\theta}\right) &= r\left(\boldsymbol{\zeta} \,|\, \mathbf{z}\right) \cdot p\left(\mathbf{z} \,|\, \boldsymbol{\theta}\right), \qquad \text{where} \\ r\left(\boldsymbol{\zeta} \,|\, \mathbf{z}\right) &= \prod_i r\left(\zeta_i \,|\, z_i\right) \end{aligned} \tag{6}$$

The posterior is correspondingly augmented as follows

$$q\left(\boldsymbol{\zeta}, \mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi}\right) = r\left(\boldsymbol{\zeta} \,|\, \mathbf{z}\right) \cdot q\left(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi}\right) \tag{7}$$

The distribution $r(\boldsymbol{\zeta} \,|\, \mathbf{z})$ is the same for both the posterior and the prior. Although the following discussion is generic and can be applied to any probabilistic model, we restrict the notation to that for variational autoencoders. Similar to Variational Autoencoders, we can write the ELBO as a sum of a KL term and an autoencoding term.

$$\mathcal{L}\left(\mathbf{z}, \boldsymbol{\phi}\right) = \underbrace{-\mathrm{KL}\left(q\left(\boldsymbol{\zeta}, \mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi}\right) \,||\, p\left(\boldsymbol{\zeta}, \mathbf{z} \,|\, \boldsymbol{\theta}\right)\right)}_{\text{KL Term}} + \underbrace{\mathop{\mathbb{E}}_{q(\boldsymbol{\zeta}, \mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi})}\left[p(\mathbf{x} \,|\, \boldsymbol{\zeta}, \boldsymbol{\theta})\right]}_{\text{Autoencoding Term}} \tag{8}$$

The gradient of the autoencoding term can be computed by sampling the latent variables $\boldsymbol{\zeta}$. This is possible due to reparametrization of the latent variable $\boldsymbol{\zeta}$, which is discussed later in this section. For the KL term, Rolfe (2016) give a very detailed explanation on computing the gradients. However, since I assume the autoregressive connections to be absent from the network, the computation of the KL term greatly simplifies. The gradients for the KL term can be written as

$$\nabla \mathrm{KL}\left(q \,||\, p\right) = \underbrace{\mathop{\mathbb{E}}_{q(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi})}\left[\nabla E_{\boldsymbol{\theta}}(\mathbf{z})\right] - \mathop{\mathbb{E}}_{p(\mathbf{z} \,|\, \boldsymbol{\theta})}\left[\nabla E_{\boldsymbol{\theta}}(\mathbf{z})\right]}_{\text{Gradients for Prior Parameters}} + \tag{9}$$

$$\underbrace{\sum_i \sum_{z_i} q\left(z_i \,|\, \mathbf{x}, \boldsymbol{\phi}\right) \log q\left(z_i \,|\, \mathbf{x}, \boldsymbol{\phi}\right) + \nabla E_{\boldsymbol{\theta}}\left(\mathop{\mathbb{E}}_{q(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi})}\left[\mathbf{z}\right]\right)}_{\text{Gradients for Posterior Parameters}} \tag{10}$$

Note. The above equation can only be written if the posterior of latent variables $\mathbf{z}$ is assumed to be such that we can write $q\left(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi}\right) = \prod_i p\left(z_i \,|\, \mathbf{x}, \boldsymbol{\phi}\right)$

The gradients for the KL term are independent of the continuous relaxations $\boldsymbol{\zeta}$, and therefore the only dependency of the gradients is in the autoencoding term, where we need to sample the latent variables

$\boldsymbol{\zeta}$. Rolfe (2016) propose a few relaxations, such as the spike-and-exponential relaxation, the spike-and-slab relaxation and the spike-and-gaussian relaxation. I only discuss the spike-and-exp relaxation as it was observed to perform the best.

4.1. Spike and Exponential Smoothing Transformation

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

$$\begin{aligned} r\left(\zeta_i \,|\, z_i = 0\right) &= \begin{cases} \infty, & \text{if } \zeta_i = 0 \\ 0, & \text{otherwise} \end{cases} \\ r\left(\zeta_i \,|\, z_i = 1\right) &= \begin{cases} \frac{\beta e^{\beta\zeta}}{e^{\beta}-1}, & \text{if } \zeta_i \in [\,0, 1\,] \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

The inverse CDF for the same is given (details in the original paper) as

$$\zeta_i = F^{-1}_{q(\zeta_i \,|\, \mathbf{x}, \boldsymbol{\phi})}(\rho_i) = \max\left\{ \frac{1}{\beta} \cdot \log\left[\left(\frac{\rho_i + q_i - 1}{q_i} \right) \cdot \left(e^{\beta} - 1 \right) \right] + 1, 0 \right\} \tag{12}$$

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

The spike-and-exponential smoothing allows us to effectively use binary latent variables with RBM priors. This approach was extended by Vahdat et al. (2018b), where in the spike-and-exp relaxation is replaced by overlapping transformation of two exponential distributions. The main advantage of using this is that exp-and-exp relaxations allow us to use Automatic Differentiation (AD) for learning the model, as opposed to the case in DVAE, where in the gradients (for the KL) need to be manually coded. The reason, as noted by the authors, is the symmetric distribution for $z_i = 0$ and $z_i = 1$. However, this is not observed with the assumption that the posterior decouples over the dimensions of the latent variables $\mathbf{z}$, as noted above.

4.2. Overlapping Transformations - Exponential and Exponential Smoothing

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

$$\begin{aligned} r\left(\zeta_i \,|\, z_i = 0\right) &= \begin{cases} \frac{\beta e^{\beta(1-\zeta)}}{e^{\beta}-1}, & \text{if } \zeta_i \in [\,0, 1\,] \\ 0, & \text{otherwise} \end{cases} \\ r\left(\zeta_i \,|\, z_i = 1\right) &= \begin{cases} \frac{\beta e^{\beta\zeta}}{e^{\beta}-1}, & \text{if } \zeta_i \in [\,0, 1\,] \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

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

$$\zeta_i = F^{-1}_{q(\zeta_i \,|\, \mathbf{x}, \boldsymbol{\phi})}(\rho_i) = -\frac{1}{\beta} \cdot \log \frac{-b + \sqrt{b^2 - 4ac}}{2} \tag{14}$$

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 $q\left(z_i \,|\, \mathbf{x}, \boldsymbol{\phi}\right)$.

Both Rolfe (2016) and Vahdat et al. (2018b) propose models having binary latent variables with an RBM prior, using smoothing transformations based on spipe-and-exp and exp-and-exp (respectively) transformations.

This work was further extended by Vahdat et al. (2018a), wherein a generic strategy was proposed to compute inverse CDF in case of overlapping distributions. They show their methods to work better than DVAE and DVAE++, and propose another overlapping distribution based on power distribution. Although I do not discuss much on their findings, it is an interesting work, exploring which I leave as a future task.

5. GumBolt Trick

As pointed out earlier, the Gumbel Softmax trick is built for factorial distributions and does not scale to RBM priors. Khoshaman and Amin (2018) point out that the reason behind this is the removal of the latent variables $\mathbf{z}$ from the inference model. Therefore, an alternate model, named as GumBolt is suggested by Khoshaman and Amin (2018) using the Gumbel-Softmax trick along with RBM priors.

Similar to gumbel softmax case, a relaxed probabilities are required to replace the prior and the posterior probabilities. The posterior probability is consistent with the reparameterization offered by the Gumbel-Softmax Trick, however, the prior probability is replaced with a proxy, which is given as

$$\widetilde{p}\left(\boldsymbol{\zeta} \,|\, \boldsymbol{\theta}\right) \quad = \quad \frac{e^{-E_{\boldsymbol{\theta}}(\boldsymbol{\zeta})}}{Z_{\boldsymbol{\theta}}} \tag{15}$$

where $Z_{\boldsymbol{\theta}}$ is the partition function for the prior probability of $\mathbf{z}$. Note, the above proxy is not really a probability distribution, as it is unnormalized, however, as the temperature (of the Gumbel-Softmax trick) limits to 0, the above proxy tends to $p\left(\mathbf{z} \,|\, \boldsymbol{\theta}\right)$. Moreover, the inconsistency of the posterior remaining a probability density function even when the temperature parameter limits to 0 is solved by replacing the posterior with a relaxed probability given as $\log \widetilde{q}\left(\boldsymbol{\zeta} \,|\, \mathbf{x}, \boldsymbol{\phi}\right) = \boldsymbol{\zeta} \cdot \log\left(q\right) + \left(1 - \boldsymbol{\zeta}\right) \cdot \log\left(1 - q\right)$, where q is the short-hand notation for $q\left(\boldsymbol{\zeta} \,|\, \mathbf{x}, \boldsymbol{\phi}\right)$. Therefore, the relaxed ELBO is written as

$$\widetilde{\mathcal{L}}(\mathbf{x}, \boldsymbol{\phi}) \quad = \quad \mathop{\mathbb{E}}_{q(\boldsymbol{\zeta} \,|\, \mathbf{x}, \boldsymbol{\phi})} \left[\log \frac{e^{-E_{\boldsymbol{\theta}}(\boldsymbol{\zeta})} p\left(\mathbf{x} \,|\, \boldsymbol{\zeta}, \boldsymbol{\theta}\right)}{\widetilde{q}\left(\boldsymbol{\zeta} \,|\, \mathbf{x}, \boldsymbol{\phi}\right)} \right] - \log Z_{\boldsymbol{\theta}} \tag{16}$$

As the temperature, $\tau \to 0$, the $\widetilde{\mathcal{L}}(\mathbf{x}, \boldsymbol{\phi}) \to \mathcal{L}\left(\mathbf{x}, \boldsymbol{\phi}\right)$. The authors prove this ELBO to be a lower bound on the actual ELBO, and therefore claim that optimizing this ELBO with annealing temperature optimizes the actual object. This approach has another advantage over the Gumbel Softmax trick, this being that the Gumbolt proxy ELBO allows the use of importance weighted samples (Burda et al., 2015), which has shown to converge better than the standard ELBO, as it is an upper bound on the traditional ELBO. Use of RBM priors and Importance Weighted samples gives Gumbolt models an upperhand over the standard VAE with the Gumbel-Softmax Trick. Moreover, the authors empirically show that Gumbolt performs better than DVAE and DVAE++, therefore proving itself to be a fecilitous approach to learning models with binary latent variable.s

6. Stochastic Block Models

A Stochastic Block Model is a generative model for graphs, where the modes are categorized into communities. Essentially, one can interpret the community of a node as the latent feature / latent representation, and the the connections between the nodes as the observed variables. The most basic Stochastic Block model partitions the set of vertices into K communities, and defines the probability of a link between the nodes of these communities. This can be inferred using an EM style framework, with the global variables being the links between nodes and the local/latent variables being the community structure.

The essential problem between this model and other models derived from the SBM is the limitation of the feature representations. The number of classes to capture the communities and the subcommunities can be

very large. Moreover, these models assume a simplistic probability function, with a fixed probability for a link between two communities. The first problem can be resolved using binary vectors as latent variables.

The second problem, however, is resolved by assuming more expressive latent variables. One such solution is presented by Kipf and Welling (2016), where in a Graph Convolution Network is used as an encoder in the variational autoencoder framework, and the reconstructed links are used for predictions. This model, however, has its own problems pertinent to the non-interpretability of the continous latent variables (normal, as in a standard VAE).

Both these problems were simultaneously tackled by Mehta and Rai (2018), an unpublished work, where in binary latent variables are used in company with gaussian variables to interpretability, while maintaining the expressiveness of continous latent variables. This work is an extension of the Non-parametric Latent Feature Relational Model (Miller et al., 2009) which uses an Indian Buffet Process (IBP) prior over the latent variables, however uses the same architecture as the standard SNM, and is therefore more difficult to train. Mehta and Rai (2018) uses the same IBP prior for the latent variables, however, the latent variables are assumed to be independent, and therefore sub-community structures cannot be efficiently generated using these priors. I propose to solve this by using an RBM prior over the binary variables, forgoing the non-parametric nature, but adding additional expressiveness to the latent variables. I name this model as Gumbolt Variational Graph Convolutional Network (Gumbolt-VGCN), attributing the name to the use of the Gumbolt Trick along with Graph Variational Autoencoders.

7. Gumbolt Variational Graph Autoencoder

As mentioned in the previous section, for each node, we have a latent representation based on a combination of two types of variables, a binary vector and a real vector. Using the same notation as Mehta and Rai (2018), we denote the binary latent variable for the n^{th} node as $\mathbf{b}_n$, where $\mathbf{b}_n \in \{0,1\}^K$ and the gaussian latent variable as $\mathbf{r}_n$ and $\mathbf{r} \in \mathbb{R}^K$. The latent representation for the n^{th} node, then, is given as $\mathbf{z}_n = \mathbf{b}_n \odot \mathbf{r}_n$. Using the VAE architecture, we model the posterior of latent variables through an encoder network based on graph convolutional networks. We first describe the structure of the decoder in the following section.

Note. Unlike Mehta and Rai (2018) and Miller et al. (2009), K, *i.e.* the number of latent dimensions is fixed, and not learnt using a non-parameteric prior

7.1. The VAE Decoder

The prior for the binary latent variables is given using an RBM machine. The variables are, therefore, decomposed into two sets, hidden and visible as $\mathbf{b} = [\,\mathbf{b}_1, \mathbf{b}_2\,]$, and are then sampled from an RBM prior as follows

$$\mathbf{b}_1, \mathbf{b}_2 \quad \sim \quad \mathrm{RBM}(\mathbf{a}_1, \mathbf{a}_2, \mathrm{W}) \tag{17}$$

The continuous variables, as mentioned before, are assumed to be sampled from a standard gaussian distribution as follows

$$\mathbf{r} \quad \sim \quad \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{18}$$

The connections between nodes are then modeled as bernoulli random variables, assumed to be independent when conditioned on the latent variables. More formally, given the latent features $\mathbf{z}$, we can sample the links between two nodes n and m as

$$\mathrm{A}_{nm} \quad = \quad \mathrm{Bernoulli}\left(\sigma\left({\mathbf{f}_n}^{\mathrm{T}}\mathbf{f}_m + c\right)\right) \tag{19}$$

where $\mathbf{f}_n$ denotes a non-linear transform of the latent representaion $\mathbf{z}_n$, *i.e.* $\mathbf{f}_n = \mathbf{f}(\mathbf{z}_n)$, and $\mathbf{f} : \mathbb{R}^K \to \mathbb{R}^{K'}$ is modeled using a neural-network.

Method	Cora		Citeseer	
	ROC	AP	ROC	AP
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-VGAE compared with baselines

7.2. The VAE Encoder

The encoder is modeled similar to that for a Graph Variational Autoencoder, using a fast, non-iterative recognition model. The parameters used to sample the latent variables $\mathbf{b}$ and $\mathbf{r}$ are encoded using a graph convolutional network (GCN). In order to simplify inference, we use the mean field assumption, which assumes conditional independence of the binary and the real latent variables. Essentially, given the feature vector for a node $\mathbf{x}_n$ and the Adjacecy matrix for the graph A, we can write the posterior using the mean-field assumption as

$$q\left(\mathbf{b}_n, \mathbf{r}_n \,|\, \mathbf{x}, \mathrm{A}, \boldsymbol{\phi}\right) \quad = \quad \prod_{k=1}^{K} q\left(b_{n,k} \,|\, \mathbf{x}_n, \mathrm{A}, \boldsymbol{\phi}\right) q\left(r_{n,k} \,|\, \mathbf{x}_n, \mathrm{A}, \boldsymbol{\phi}\right) \tag{20}$$

For consistency with the prior, we model the probabilities $q\left(b_{n,k} \,|\, \mathbf{x}, \mathrm{A}\right)$ and $q\left(r_{n,k} \,|\, \mathbf{x}, \mathrm{A}\right)$ using the bernoulli and the gaussian (respectively) distributions. The parameters for these distributions are generated, as noted above, using a GCN. We write this as

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

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

In order to infer the parameters, stochastic gradient variational bayes (SGVB) is used. Since this does not support binary (rather, discrete) latent variables, we use the Gumblt Probabilty proxy to compute the gradients ofthe KL term. The samples from the prior are generated using Persistive Contrastive Divergence. This allows us to efficiently train the model, while affording the greater expressive power of RBM priors in a generative setting.

8. Experiments

Due to the shortage of time, I was only able to conduct limited experiments, with visually analyzing the reconstruction and sampling of MNIST images using Gumbolt-VAE, and performing some experiments with Gumbolt-VGAE.

The reconstructed and sampled images using Gumbolt-VAE are shown in figure 2 and 3.

The results of the Gumbolt-VGAE compared to baselines for link prediction on datasets Citeseer and Cora are shown in table 1

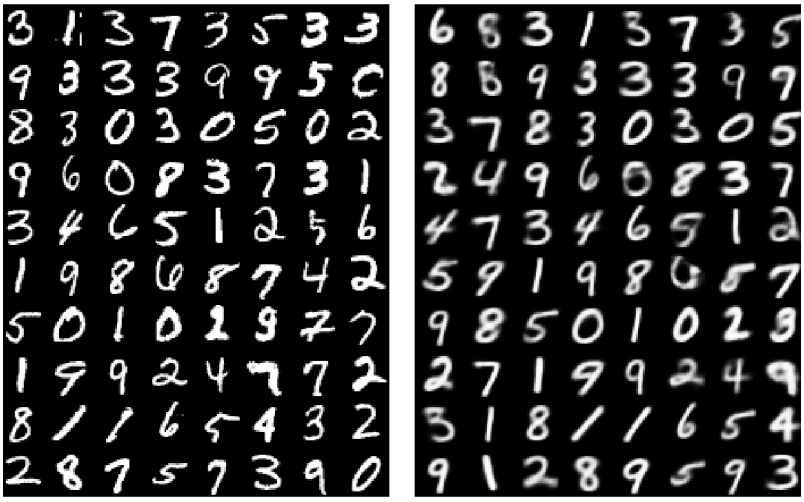


Figure 2: Reconstruction Plot for Gumbolt-Vae

Figure 3: Sampling Plot for Gumbolt-Vae

References

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance Weighted Autoencoders. *ArXiv e-prints*, September 2015.

Dominik Csiba and Peter Richtárik. Importance sampling for minibatches. *CoRR*, abs/1602.02283, 2016. URL `http://arxiv.org/abs/1602.02283`.

Michael Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. *CoRR*, abs/1805.08498, 2018. URL `http://arxiv.org/abs/1805.08498`.

Prem K Gopalan, Sean Gerrish, Michael Freedman, David M. Blei, and David M. Mimno. Scalable inference of overlapping communities. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2249–2257. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4573-scalable-inference-of-overlapping-communities.pdf`.

E. Jang, S. Gu, and B. Poole. Categorical Reparameterization with Gumbel-Softmax. *ArXiv e-prints*, November 2016.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. NIPS'13, pages 315–323, USA, 2013. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999611.2999647`.

A. H. Khoshaman and M. H. Amin. GumBolt: Extending Gumbel trick to Boltzmann priors. *ArXiv e-prints*, May 2018.

D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

T. N. Kipf and M. Welling. Variational Graph Auto-Encoders. *ArXiv e-prints*, November 2016.

C. J. Maddison, A. Mnih, and Y. Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *ArXiv e-prints*, November 2016.

Nikhil Mehta and Piyush Rai. Stochastic blockmodels meet graph neural networks. 2018.

Kurt T. Miller, Thomas L. Griffiths, and Michael I. Jordan. Nonparametric latent feature models for link prediction. In *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, NIPS'09, pages 1276–1284, USA, 2009. Curran Associates Inc. ISBN 978-1-61567-911-9. URL `http://dl.acm.org/citation.cfm?id=2984093.2984237`.

C. A. Naesseth, F. J. R. Ruiz, S. W. Linderman, and D. M. Blei. Reparameterization Gradients through Acceptance-Rejection Sampling Algorithms. *ArXiv e-prints*, October 2016.

John Paisley, David M. Blei, and Michael I. Jordan. Variational bayesian inference with stochastic search. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1363–1370, USA, 2012. Omnipress. ISBN 978-1-4503-1285-1. URL `http://dl.acm.org/citation.cfm?id=3042573.3042748`.

Dmytro Perekrestenko, Volkan Cevher, and Martin Jaggi. Faster coordinate descent via adaptive importance sampling. *CoRR*, abs/1703.02518, 2017. URL `http://arxiv.org/abs/1703.02518`.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *AISTATS*, 2014.

J. T. Rolfe. Discrete Variational Autoencoders. *ArXiv e-prints*, September 2016.

F. J. R. Ruiz, M. K. Titsias, and D. M. Blei. The Generalized Reparameterization Gradient. *ArXiv e-prints*, October 2016.

A. Vahdat, E. Andriyash, and W. G. Macready. DVAE#: Discrete Variational Autoencoders with Relaxed Boltzmann Priors. *ArXiv e-prints*, May 2018a.

A. Vahdat, W. G. Macready, Z. Bian, A. Khoshaman, and E. Andriyash. DVAE++: Discrete Variational Autoencoders with Overlapping Transformations. *ArXiv e-prints*, February 2018b.

Chong Wang, Xi Chen, Alexander J Smola, and Eric P Xing. Variance reduction for stochastic gradient optimization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 181–189. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5034-variance-reduction-for-stochastic-gradient-optimization.pdf`.