Decoupling Sparsity and Smoothness in Dirichlet Belief Networks

Yaqiong Li[1], Xuhui Fan[2]✉, Ling Chen[1], Bin Li[3], and Scott A. Sisson[2]

[1] Australian Artificial Intelligence Institute, University of Technology, Sydney
yaqiong.li@student.uts.edu.au,ling.chen@uts.edu.au
[2] UNSW Data Science Hub, and School of Mathematics and Statistics,
University of New South Wales
{xuhui.fan,scott.sisson}@unsw.edu.au
[3] Shanghai Key Laboratory of IIP, School of Computer Science, Fudan University
libin@fudan.edu.cn

Abstract. The Dirichlet Belief Network (DirBN) has been proposed as a promising deep generative model that uses Dirichlet distributions to form layer-wise connections and thereby construct a multi-stochastic layered deep architecture. However, the DirBN cannot simultaneously achieve both *sparsity*, whereby the generated latent distributions place weights on a subset of components, and *smoothness*, which requires that the posterior distribution should not be dominated by the data. To address this limitation we introduce the sparse and smooth Dirichlet Belief Network (ssDirBN) which can achieve both sparsity and smoothness simultaneously, thereby increasing modelling flexibility over the DirBN. This gain is achieved by introducing binary variables to indicate whether each entity's latent distribution at each layer uses a particular component. As a result, each latent distribution may use only a subset of components in each layer, and smoothness is enforced on this subset. Extra efforts on modifying the models are also made to fix the issues which is caused by introducing these binary variables. Extensive experimental results on real-world data show significant performance improvements of ssDirBN over state-of-the-art models in terms of both enhanced model predictions and reduced model complexity.

Keywords: Dirichlet Belief Networks · Markov chain Monte Carlo · Sparsity.

1 Introduction

The Dirichlet Belief Network (DirBN) [20] was recently proposed as a promising deep probabilistic framework for learning *interpretable* hierarchical latent distributions for entities (or objects). To date, DirBN has been successfully implemented in two application areas: (1) topic structure learning [20], where the entities represent topics and the entities' latent distributions describe the topic's vocabulary distributions; and (2) relational modelling [4, 12, 5], where the entities represent individuals and the latent distributions characterise an individual's membership distribution over community structures. By constructing a deep architecture for the latent distributions, the DirBN can effectively model high-order dependency between topic-vocabulary distributions (for topic models) and individual's membership distributions (for relational models).

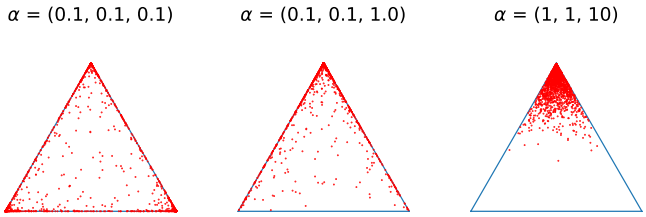


Fig. 1: Small Dirichlet concentration parameters generate sparse latent distributions. 1 500 samples (red dots) generated from a 3-dimensional Dirichlet distribution are shown on the 2-dimensional unit simplex, $x_1 + x_2 + x_3 = 1$, with different concentration parameters $\boldsymbol{\alpha}$. When $\boldsymbol{\alpha}$ is small (left and middle panels), most samples reside on vertices or edges, placing most mass on one or two dimensions. When $\boldsymbol{\alpha}$ is not small (right panel), most samples lie inside the triangle, placing mass on all three dimensions.

While promising, DirBN currently has some structural limitations which reduce their modelling flexibility. One limitation is that the length of each entity's latent distribution is restricted to be the same over all entities and all layers. As constructed, this restriction can reveal inadequate modelling flexibility in the DirBN when entities are related to different subsets of components in different layers (e.g. when individuals belong to different communities for different layers in the relational modelling setting). Since the latent distributions are linearly scaled (by Gamma-distributed variables) when being propagated into each subsequent layer, the resulting changes in the latent distributions can be too slow to adequately model the rapid changes inherent in the data.

A second limitation of DirBN is that it is unable to achieve the desirable properties of sparsity and smoothness simultaneously. Sparsity is achieved when the generated latent distributions place large weight on a subset of components – for the Dirichlet distribution, this occurs when the concentration parameters approach zero (see Fig. 1). In this case, however, the resulting posterior distribution over the latent distributions would be less smooth across layers as the empirical counts will then dominate the posterior distribution. Typically (though not exclusively) the posterior distribution is expected to be smooth, so as to reduce sensitivity to rarely-occurring latent distribution components.

In order to resolve these issues, we propose a sparse and smooth Dirichlet Belief Network (ssDirBN), which introduces binary variables into the layer-wise connections of the DirBN. In particular, each binary variable $b_{ik}^{(l)}$, which is generated by a Bernoulli distribution with entity-specific parameter, determines whether entity i's latent distribution at layer l uses component k ($b_{ik}^{(l)} = 1$) or not ($b_{ik}^{(l)} = 0$). Under this representation, sparsity is achieved through the Bernoulli distribution that generates the binary variable $b_{ik}^{(l)}$, flexibly permitting the latent distributions of different layers to solely focus on different subsets of components. Smoothness can then be enforced over those components with non-zero $b_{ik}^{(l)}$ through the Dirichlet concentration parameters. In this manner sparsity and smoothness are decoupled, and the benefits of both may be simultaneously obtained.

To ensure latent distributions to be defined appropriately and enable efficient posterior inference, we make two further modifications on the model: (1) fixing $b_{iK}^{(l)} = 1$, so that the last component K is certain to be propagated into the next layer, which can

guarantee latent distributions be defined on at least one component; (2) letting those components with $b_{ik}^{(l)} = 0$ be propagated into component K, which can satisfy the specific condition (specified in the last paragraph of Section 2) for efficient Gibbs sampling algorithms on the membership distributions.

We explore the effectiveness of the ssDirBN in context of relational models, which use multi-stochastic layered latent distributions to model individual's membership distributions over communities. In this setting, the ssDirBN permits individuals to belong to different subsets of communities within different layers, and can thereby obviate placing unnecessary small probability masses on unrelated communities. Our experimental results on real-world data show significant performance improvements of ssDirBN over DirBN and other state-of-the-art models, in terms of reduced model complexity and improved link prediction performance. Similar to DirBN that can be considered as a self-contained module [20], the ssDirBN can be flexibly combined with alternative emission models and be implemented in these applications such as topic data, collaborative filtering data, etc.

2 Preliminary Knowledge

We first give a brief review on the DirBN model, where we use N to denote the number of entities (or number of topics in topic modeling) in each layer, K to denote the number of components in the entity's latent distributions and L to denote the number of layers. In general, DirBN assumes each entity has latent distributions $\boldsymbol{\pi}_i^{(l-1)}$ at layer $l-1$ and uses Dirichlet distributions to generate entities' latent distribution at layer l, with the concentration parameters being the linear sum of entities' latent distributions at layer $l-1$. Within the relational modelling setting, $\{\boldsymbol{\pi}_i^{(l)}\}_{l=1}^{L}$ represent entity i's membership distributions over K communities at L layers. The generative process of propagating the membership distributions $\{\boldsymbol{\pi}_j^{(l-1)}\}_j$ to $\boldsymbol{\pi}_i^{(l)}$ at layer l can be briefed as follows:

1. $\beta_{ji}^{(l)} \sim \text{Gam}(c_j, d), \forall i, j = 1, \ldots, N$
2. $\boldsymbol{\pi}_i^{(l)} \sim \text{Dir}(\sum_j \beta_{ji}^{(l)} \boldsymbol{\pi}_j^{(l-1)}), \forall i = 1, \ldots, N, l = 1, \ldots, L$

where $\text{Gam}(c, d)$ is the Gamma distribution with mean c/d and variance c/d^2. $\beta_{ji}^{(l)}$ represents the information propagation coefficient from entity j at layer $l-1$ to entity i in at layer l, c_j, d are the hyper-parameters. After generating entity i's membership distribution at layer L, [4] uses counting vectors $\boldsymbol{m}_i^{(L)}$, which is sampled from Multinomial distribution with $\boldsymbol{\pi}_i^{(L)}$ as event probabilities, and community compatibility matrix to form probability function for generating the entity-wise relations.

DirBN is mostly inferred through Markov chain Monte Carlo (MCMC) methods. To enable efficient Gibbs sampling algorithm for DirBN, we note that the probability density function of $\boldsymbol{\pi}_i^{(l)}$ is written as:

$$P(\boldsymbol{\pi}_i^{(l)}|-) = \frac{\Gamma(\sum_k \sum_j \beta_{ji}^{(l)} \pi_{jk}^{(l-1)})}{\prod_k \Gamma(\sum_j \beta_{ji}^{(l)} \pi_{jk}^{(l-1)})} \prod_k (\pi_{ik}^{(l)})^{\sum_j \beta_{ji}^{(l)} \pi_{jk}^{(l-1)} - 1} \tag{1}$$

where $(-)$ refers to the set of conditional variables related to $\boldsymbol{\pi}_i^{(l)}$ and $\Gamma(\cdot)$ is the Gamma function. As $\{\pi_{jk}^{(l-1)}\}_j$ appear in the Gamma function, the prior and posterior distributions of $\boldsymbol{\pi}_i^{(l-1)}$ are not conjugate and it is difficult to implement efficient Gibbs sampling for $\boldsymbol{\pi}_i^{(l-1)}$. A strategy of first upward propagating latent counts and then downward sampling variables has been developed in [20] to address this issue, which is detailed below.

Upward propagating latent counts W.l.o.g., we assume the observation at layer l is the counts $\boldsymbol{m}_i^{(l)}$, which is obtained through Multinomial distribution with $\boldsymbol{\pi}_i^{(l)}$ as event probabilities. We may first integrate $\boldsymbol{\pi}_i^{(l)}$ out and obtain the likelihood term of the latent counts $\boldsymbol{m}_i^{(l)}$ as:

$$P(\{m_{ik}^{(l)}\}_k|\{\pi_{jk}^{(l-1)}\}_{j,k}) \propto \prod_k \frac{\Gamma(\sum_j \beta_{ji}^{(l)}\pi_{jk}^{(l-1)} + m_{ik}^{(l)})}{\Gamma(\sum_j \beta_{ji}^{(l)}\pi_{jk}^{(l)})} \tag{2}$$

The r.h.s. in Equation (2) can be augmented through a random counts $y_{ik}^{(l)}$ from the Chinese Restaurant Table (CRT) distribution (i.e. $y_{ik}^{(l)} \sim \text{CRT}(m_{ik}^{(l)}, \sum_j \beta_{ji}^{(l)}\pi_{jk}^{(l)})$) as:

$$P(\{y_{ik}^{(l)}\}_k, \{m_{ik}^{(l)}\}_k|\{\pi_{ik}^{(l-1)}\}_k) \propto \prod_k [(\sum_j \beta_{ji}^{(l)}\pi_{jk}^{(l-1)})^{y_{ik}^{(l)}}(m_{ik}^{(l)} - y_{ik}^{(l)})!]$$

By further distributing the 'derived' count $y_{ik}^{(l)}$ into the entities at layer $l-1$ through a Multinomial distribution as: $(h_{1ik}^{(l)}, \ldots, h_{Nik}^{(l)}) \sim \text{Multi}(y_{ik}^{(l)}; \frac{\{\pi_{jk}^{(l-1)}B_{ji}^{(l-1)}\}_j}{\sum_j \beta_{ji}^{(l)}\pi_{jk}^{(l)}})$ and the terms associated with $\{\boldsymbol{\pi}_i^{(l-1)}\}_i$ are abstracted as:

$$P(\{h_{jik}^{(l-1)}\}_{j,k}|\{\pi_{jk}^{(l-1)}\}_k) \propto \prod_k (\pi_{jk}^{(l-1)})^{\sum_j h_{jik}^{(l-1)}}$$

The latent counts $\boldsymbol{m}_i^{(l-1)} = (\sum_j h_{ji1}^{(l-1)}, \ldots, \sum_j h_{jiK}^{(l-1)})$ can be regarded as a random draw from a Multinomial distribution, with $\boldsymbol{\pi}_i^{(l-1)}$ as event probabilities.

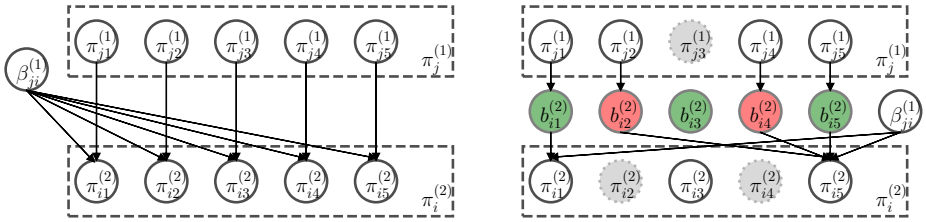
Downward sampling variables After the counts are propagated to entity i at each layer l as $m_{ik}^{(l)}$, the posterior distribution of $\boldsymbol{\pi}_i^{(l)}$ follows as:

$$\boldsymbol{\pi}_i^{(l)} \sim \text{Dir}(\sum_j \beta_{ji}^{(l)}\boldsymbol{\pi}_j^{(l-1)} + \boldsymbol{m}_i^{(l-1)})$$

3 Sparse and Smooth Dirichlet Belief Networks

3.1 Generative Process

ssDirBN aims at enabling each entity's latent distribution at each layer to be defined on individual subsets of components and thus simultaneously obtain the benefits of sparsity and smoothness of the Dirichlet distribution. Given N entities' latent distributions $\{\boldsymbol{\pi}_i^{(l-1)}\}_{i=1}^N$ at layer $l-1$, we use the following method to generate entity i's latent distribution $\boldsymbol{\pi}_i^{(l)}$ at layer l:



(a) DirBN (b) ssDirBN

Fig. 2: Example visualisations on propagating latent distributions of $\boldsymbol{\pi}_j^{(1)}$ at layer 1 to $\boldsymbol{\pi}_i^{(2)}$ at layer 2. In DirBN (left panel), all the components in $\boldsymbol{\pi}_j^{(1)}$ are one-to-one propagated to their corresponding components in $\boldsymbol{\pi}_i^{(2)}$. In ssDirBN (right panel), binary variables are inserted between $\boldsymbol{\pi}_j^{(1)}$ and $\boldsymbol{\pi}_i^{(2)}$. Red entities of $b_{ik}^{(2)}$ represent "$b_{ik}^{(2)} = 0$" and in this case, $\pi_{jk}^{(1)}$ will be propagated to $\pi_{i5}^{(2)}$, which is the last component of $\boldsymbol{\pi}_i^{(2)}$. Green $b_{ik}^{(2)}$ entities represent that $b_{ik}^{(2)} = 1$ and in this case, $\pi_{jk}^{(1)}$ will be propagated to $\pi_{ik}^{(2)}$. Grey dotted entities represent non-existing components. Since $\pi_{j3}^{(1)}$ is not existing, $\boldsymbol{\pi}_j^{(1)}$ can not propagate component 3 to $\boldsymbol{\pi}_i^{(2)}$ (other entities with existing component 3 will propagate to it.).

1. $\beta_{ji}^{(l)} \sim \text{Gam}(c_j, d), \forall i, j = 1, \ldots, N$
2. $\omega_i^{(l)} \sim \text{Beta}(\gamma, s), b_{iK}^{(l)} = 1, \{b_{ik}^{(l)}\}_{k=1}^{K-1} \sim \text{Bernoulli}(\omega_i^{(l)})$
3. $\boldsymbol{\pi}_i^{(l)} \sim \text{Dir}\left(\sum_j \beta_{ji}^{(l)} (\boldsymbol{b}_i^{(l)} \cdot \boldsymbol{\pi}_j^{(l-1)} + \boldsymbol{e}_K \cdot (\sum_k \pi_{jk}^{(l-1)} \delta_{b_{ik}^{(l)}=0}))\right), \forall i = 1, \ldots, N$

where $\boldsymbol{e}_K = [0, \ldots, 0, 1]$ is a K-length vector with the last element 1 and 0 elsewhere; the two multiplication operations $\cdot$ in step 3 represents element-wise vector multiplication and scalar and vector multiplication respectively; c_j, d, γ, s are the hyper-parameters and we set Gamma priors on them as: $c_j \sim \text{Gam}(c_0/K, d_0)$,$d \sim \text{Gam}(e_0, f_0)$, $\gamma, s \sim \text{Gam}(g_0, h_0)$.

In this generative process, step 1 generates layer l's entity-wise information propagation coefficient $\beta_{ji}^{(l)}$, which is same as in DirBN and represents the information propagation coefficient from entity j's latent distribution at layer $l-1$ to that of entity i at layer l.

Step 2 generates a component including variable $\omega_i^{(l)}$ and a subsequent K-length binary vector $\boldsymbol{b}_i^{(l)} \in \{0, 1\}^{1 \times K}$ for entity i at layer l. When larger values of $\omega_i^{(l)}$ encourage more "1" entries in $\boldsymbol{b}_i^{(l)}$, the case of $b_{ik}^{(l)} = 1$ denotes that entity i's latent distribution at layer l includes component k and vice versa. That is, $\boldsymbol{b}_i$ specifies a small simplex through its "1" entries. An exception is the component K, for which we fix $b_{iK}^{(l)}$ as $b_{iK}^{(l)} = 1$ to make sure $\boldsymbol{\pi}_i^{(l)}$ is well defined ($\boldsymbol{\pi}_i^{(l)}$ will be problematic if $b_{i,k}^{(l)} = 0$ for $k = 1, \ldots, K$). $\boldsymbol{b}_i^{(l)}$ determines the subset of components for entity i's latent distribution at layer l, i.e., $|\boldsymbol{\pi}_i^{(l)}| = \sum_k b_{ik}^{(l)}$.

Step 3 also uses Dirichlet distribution to generate $\boldsymbol{\pi}_i^{(l)}$ at layer l. The related concentration parameter is a linear sum of the entities' components weight at layer $l-1$. For component k, its weight propagation would be proceeded differently based on the value

of $b_{ik}^{(l)}$: (1) when $b_{ik}^{(l)} = 1$, $\pi_{jk}^{(l-1)}$ will be added to component k of the concentration parameters; (2) when $b_{ik}^{(l)} = 0$, $\pi_{jk}^{(l-1)}$ will be added to component K of the concentration parameters. Fig. 2 shows an example to visualise the propagation of latent distributions in DirBN and ssDirBN respectively.

Our ssDirBN has the following advantages when compared to DirBN:

- **Flexible subsets of components** By introducing the binary variables, the latent distributions for different entities at different layers can be defined on different subsets of components. The model complexity is reduced as fewer components are involved. Also, the flexible usage of components may help each latent distribution focus on closely related components without assigning weights to unrelated ones.
- **Flexible weight ratios between components** Recall that the ratios of component weights are unchanged when propagated from the current layer to the next layer. Thus, the concentration parameters of Dirichlet distributions should also follow these ratios generally. In ssDirBN, since we allow some components to be non-existing, the ratios of components can thus change greatly during the layer-wise connections, which enhances the representation capability of the model.
- **Decoupling sparsity and smoothness** The sparsity in our ssDirBN is controlled by the Bernoulli parameter ω_i for entity i, with component k retained only if $b_{ik}^{(l)} = 1$. The smoothing effect is placed on the remaining components and controlled through the linear coefficient $\beta_{ji}^{(l)}$. In this way, the sparsity and smoothness are decoupled and we can obtain the benefits of both properties at the same time.

3.2 Necessity of fixing $b_{iK}^{(l)} = 1$

Fixing $b_{iK}^{(l)} = 1$ $(\forall i, l)$ and making component k propagate to component K when $b_{ik}^{(l)} = 0$ $(\forall k)$ are key steps to guarantee the feasibility of upward count propagation method in ssDirBN. Recall that we have $\sum_j \beta_{ji}^{(l)} = \sum_k \sum_j \beta_{ji} \pi_{jk}^{(l)}$ to ensure that generated counts follow a Multinomial distribution. If we directly introduce the binary variable $b_{ik}^{(l)}$, which makes $b_{ik}^{(l)} \sim \text{Bernoulli}(\omega_i)(\forall k)$, we have

$$P(\{h_{jik}^{(l)}\}_j | \{\pi_{jk}^{(l-1)}\}_k) \propto [q_i^{(l)}]^{\sum_k \sum_j \beta_{ji} b_{ik}^{(l)} \pi_{jk}^{(l)}} \prod_k (\pi_{jk}^{(l-1)})^{h_{jik}^{(l)}}$$

That is, the counts of $\{h_{jik}^{(l)}\}_j$ *cannot* form a Multinomial distribution. However, we can still obtain $\sum_j \beta_{ji}^{(l)} = \sum_k \sum_j \beta_{ji} b_{jk}^{(l)} \pi_{jk}^{(l)}$ in ssDirBN, which enables the upward count propagation.

4 Related Work

In addition to the DirBN variants mentioned in the introduction, ssDirBN is also closely related to Gamma Belief Networks (GBN) [22], which is another multi-stochastic layered deep generative model. Instead of Dirichlet distributions, GBN used Gamma distributions to propagate scalar variables between layers and was the first to develop upward latent counts propagation and downward variable sampling method for model inference.

Applications of GBN and the related inference technique have been observed in natural language modelling [7], Dynamic Systems [14, 8, 17, 13] and even variational auto-encoder methods [18]. GBN does not enjoy the unique sparsity property of Dirichlet distribution and cannot be used to model latent distributions.

The basic idea of our ssDirBN is inspired by the sparse topic models (sparseTM) [16, 2]. Compared with our ssDirBN, sparseTM places binary variables for all the components of the Dirichlet distribution. However, as a shallow model, sparseTM cannot model the complex entity-wise dependencies. Our usage of binary variables may also be similar to the techniques of Bayesian-dropout [6], which uses binary variables to decide whether or not to propagate the corresponding neuron to the next layer. In ssDirBN, we propagate the "neuron" to the last component when the binary variable equals to 0, rather than directly discarding it.

5 ssDirBN for relational modelling

We apply our ssDirBN in the setting of relational modelling, which focuses on constructing multi-stochastic layered membership distributions over communities for entities. The detail generative process is expressed as follows:

1. $\boldsymbol{\pi}_i^{(1)} \sim \text{Dirichlet}(\boldsymbol{\alpha})$;
2. For $l = 2, \ldots, L$
   - $\beta_{ji}^{(l-1)} \begin{cases} \sim \text{Gam}(c_i, d),\ j \in \{j : R_{ji} = 1\} \cup \{i\}; \\ = 0, \qquad\qquad \text{otherwise}; \end{cases}$;
   - $\omega_i^{(l)} \sim \text{Beta}(\gamma, s), b_{iK}^{(l)} = 1, \{b_{ik}^{(l)}\}_{k=1}^{K-1} \sim \text{Bernoulli}(\omega_i^{(l)})$
   - $\boldsymbol{\pi}_i^{(l)} \sim \text{Dir}\left(\sum_j \beta_{ji}^{(l)} (\boldsymbol{b}_i^{(l)} \cdot \boldsymbol{\pi}_j^{(l-1)} + \boldsymbol{e}_K \cdot (\sum_k \pi_{jk}^{(l-1)} \delta_{b_{ik}^{(l)}=0}))\right)$
3. $M_i \sim \text{Poisson}(M), (X_{i1}, \ldots, X_{iK}) \sim \text{Multi}(M_i; \pi_{i1}^{(L)}, \ldots, \pi_{iK}^{(L)})$;
4. $\Lambda_{k_1 k_2} \sim \text{Gam}(k_\Lambda, \frac{1}{\theta_\Lambda})$;
5. $Z_{ij,k_1 k_2} \sim \text{Poisson}(X_{ik_1} \Lambda_{k_1 k_2} X_{jk_2})$;
6. $R_{ij} = \mathbf{1}(\sum_{k_1,k_2} Z_{ij,k_1 k_2} > 0)$.

In this generative process, $\boldsymbol{\alpha}$ in line 1 represents the concentration parameter in the Dirichlet distribution in generate all the entities' latent distribution at layer 1; line 2 represents our proposed ssDirBN structure, which can generate sparse and smooth latent distributions at layer L; line 3 generates latent count variable $(X_{i1}, \ldots, X_{iK})$ for entity i's latent distribution $\boldsymbol{\pi}_i^{(L)}$ at layer L; $\Lambda_{k_1 k_2}$ in line 4 is a community compatibility parameter such that a larger value of $\Lambda_{k_1 k_2}$ indicates a larger possibility of generating the links between community k_1 and community k_2; and $Z_{ij,k_1 k_2,t}$ is a community-to-community latent integer for each relation R_{ij}.

It is noted that, through the Multinomial distributions with $\boldsymbol{\pi}_i^{(L)}$ as event probabilities, $\boldsymbol{X}_i$ can be regarded as an estimator of $\boldsymbol{\pi}_i^{(L)}$. Since the sum also follows a Poisson distribution as $M_i \sim \text{Poisson}(M)$, according to the Poisson-Multinomial equivalence, each X_{ik} is equivalently distributed as $X_{i,k} \sim \text{Poisson}(M\pi_{ik}^{(L)})$. Therefore, both the prior distribution for generating X_{ik} and the likelihood based on X_{ik} are Poisson distributions. We may form feasible categorical distribution on its posterior inference. This

trick is inspired by the recent advances in data augmentation and marginalisation techniques [4], which allows us to implement posterior sampling for X_{ik} efficiently.

The counts $\boldsymbol{X}_i$ lead to the generation of the $K \times K$ integer matrix Z_{ij,k_1k_2}. Based on the Bernoulli-Poisson link function [3, 21], the observed R_{ij} is mapped to the latent Poisson count random variable matrix $\boldsymbol{C}_{ij}$. It is shown in [4] that $\{C_{ij,k_1k_2}\}_{k_1,k_2} = 0$ if $R_{ij} = 0$. That is, only the non-zero links are involved during the inference for $\boldsymbol{C}_{ij,k_1k_2}$, which largely reduces the computational complexity, especially for large and sparse dynamic relational data. That is, since we use the Poisson-Bernoulli likelihood in modeling the relations, the computational cost of our model scales to the number of positive links.

5.1 Inference

We adopt the Markov chain Monte Carlo (MCMC) algorithm to iteratively sample the random variables from their posterior distributions. The latent conditional distributions of random variables we are approximating are: $\{\boldsymbol{\pi}_i^{(l)}\}_{i,l}$, which involves upward propagating the counting variable to each layer and downward sampling the variables of $\boldsymbol{\pi}$, the binary variable $b_{ik}^{(l)}$; $\boldsymbol{\alpha}$, which is the concentration parameters in generating the latent distributions at layer 1; scaling parameter M, which controls the latent count variables at layer L; $\{\Lambda_{k_1k_2}\}_{k_1,k_2}$, which denotes the compatibility values between community k_1 and community k_2; $\{Z_{ij,k_1k_2}\}_{i,j,k_1,k_2}$, which represents the latent integer variable for the relation from entity i to entity j from community k_1 to community k_2.

Upward propagating latent counts Using similar techniques of DirBN, we first upward propagate entity i's latent counts $\boldsymbol{m}_i^{(l)}$ at layer l to all the entities at layer $l-1$ via the following steps:

- sample 'derived latent counts' $y_{ik}^{(l)}$ and Beta random variable $q_i^{(l)}$ as

$$y_{ik}^{(t)} \sim \text{CRT}(m_{ik}^{(s)}, \sum_j \beta_{ji}\pi_{jk}^{(l-1)}), q_i^{(s)} \sim \text{Beta}(\sum_j \beta_{ji}^{(l)}, \sum_k m_{ik}^{(s)}),$$
$$y_{iK}^{(t)} \sim \text{CRT}(m_{iK}^{(s)}, \sum_j \beta_{ji}\pi_{jK}^{(l-1)} + \sum_{j,k} \beta_{ji}\pi_{jk}^{(l-1)}\delta_{b_{ik}=0});$$

- distribute count $y_{ik}^{(l)}$ to the entities at layer $l-1$ as follows:

$$\{h_{ijk}^{(l-1)}\}_j \sim \text{Multi}(y_{ik}^{(l)}; \frac{\{\pi_{jk}^{(l-1)}\beta_{ji}^{(l)}\}_j}{\sum_j \beta_{ji}\pi_{jk}^{(l)}}), \qquad \text{if } k = 1, \ldots, K-1;$$
$$\{\{h_{ijK}^{(l-1)}\}_j, \{h_{ijk}^{(l-1)}\}_{(i,k):b_{ik}^{(l)}=0}\} \sim \text{Multi}(y_{iK}^{(l)}; \frac{\{\{\pi_{jK}^{(l-1)}\beta_{ji}^{(l)}\}_j, \{\pi_{jk}^{(l-1)}\beta_{ji}^{(l)}\}_{(j,k):b_{jk}^{(l)}=0}\}}{\sum_j \beta_{ji}\pi_{jK}^{(l)} + \sum_{(j,k):b_{jk}^{(l)}=0} \pi_{jk}^{(l-1)}\beta_{ji}^{(l)}}), \text{otherwise.}$$

- collect all the latent counts propagated to entity i at layer $l-1$ as $m_{ik}^{(l-1)} = \sum_j h_{jik}^{(l-1)}$.

Downward sampling $\boldsymbol{\pi}_i^{(l)}$ Given the upward propagated latent counts $\boldsymbol{m}_i^{(l)}$, entity i's latent distribution at layer l can be sampled as:

$$\boldsymbol{\pi}_i^{(l)} \sim \text{Dir}(\sum_j \beta_{ji}^{(l)}(\boldsymbol{b}_i^{(l)} \cdot \boldsymbol{\pi}_j^{(l-1)} + \boldsymbol{e}_K \cdot (\sum_k \pi_{jk}^{(l-1)}\delta_{b_{ik}^{(l)}=0})) + \boldsymbol{m}_i^{(l)})$$

Sampling $b_{ik}^{(l)}$ We integrate out $\omega_i^{(l)}$ to sample $b_{ik}^{(l)}$. Since new values of $b_{ik}^{(l)}$ will lead to new $\boldsymbol{\pi}_i^{(l)}$ (the length of $\boldsymbol{\pi}_i^{(l)}$ is different), we also need to generate new value for $\boldsymbol{\pi}_i^{(l)}$ when calculating the acceptance ratio. Given the current binary value of $b_{ik}^{(l)}$, we use the generative process to generate a new latent distribution $\boldsymbol{\pi}_i^{(l,*)}$ based on the opposite value of $b_{ik}^{(l)}$ and then accept the opposite value of $b_{ik}^{(l)}$ and $\boldsymbol{\pi}_i^{(l,*)}$ with a ratio of $\min(1,\alpha)$, where α is defined as:

$$\alpha = \frac{P(b_{ik}^{(l,*)}|b_{i,/k}^{(l)},\gamma,s)}{P(b_{ik}^{(l)}|b_{i,/k}^{(l)},\gamma,s)} \cdot \frac{P(\{\boldsymbol{\pi}_j^{(l-1)},\boldsymbol{\pi}_j^{(l+1)}\}_j|\boldsymbol{\pi}_i^{(l,*)},-)}{P(\{\boldsymbol{\pi}_j^{(l-1)},\boldsymbol{\pi}_j^{(l+1)}\}_j|\boldsymbol{\pi}_i^{(l)},-)} \tag{3}$$

Sampling $\{\beta_{ji}^{(l)}\}_{j,i,l}$ For For $j \in \{j : R_{ji} \neq \{i\}$, the prior for $\beta_{ji}^{(l)}$ is Gam(c_i, d), the posterior distribution is

$$\beta_{ji}^{(l)} \sim \text{Gam}(c_j + \sum_k h_{jik}^{(l)}, d - \log q_i^{(l)}) \tag{4}$$

Sampling $\{X_{ik}\}_{i,k}$**:** We have $M_i \sim \text{Poisson}(M)$,

$$(X_{i1},\ldots,X_{iK}) \sim \text{Multi}(M_i;\pi_{i1}^{(L)},\ldots,\pi_{iK}^{(L)}) \overset{d}{=} X_{ik} \sim \text{Poisson}(M\pi_{ik}^{(L)}), \forall k.$$

Both the prior distribution for generating X_{ik} and the likelihood parametrised by X_{ik} are Poisson distributions. The full conditional distribution of X_{ik} (assuming $z_{ii,\cdot\cdot} = 0, \forall i$) is then

$$P(X_{ik}|M,\boldsymbol{\pi},\boldsymbol{\Lambda},\boldsymbol{Z}) \propto \frac{\left[M\pi_{ik}^{(L)} e^{-\sum_{j\neq i,k_2} X_{jk_2}(\Lambda_{kk_2}+\Lambda_{k_2k})}\right]^{X_{ik}}}{X_{ik}!} (X_{ik})^{\sum_{j_1,k_2} Z_{ij_1,kk_2} + \sum_{j_2,k_1} Z_{j_2 i,k_1 k}}. \tag{5}$$

This follows the form of Touchard polynomials, where $1 = \frac{1}{e^x T_n(x)} \sum_{k=0}^{\infty} \frac{x^k k^n}{k!}$ with $T_n(x) = \sum_{k=0}^{n} \{{n \atop k}\} x^k$ and where $\{{n \atop k}\}$ is the Stirling number of the second kind. A draw from (5) is then available by comparing a Uniform$(0,1)$ random variable to the cumulative sum of $\{\frac{1}{e^x T_n(x)} \cdot \frac{x^k k^n}{k!}\}_k$.

Sampling $\{Z_{ij,k_1k_2}\}_{i,j,k_1,k_2}$ We first sample $Z_{ij,\cdot\cdot}$ from a Poisson distribution with positive support:

$$Z_{ij,\cdot\cdot} \sim \text{Poisson}_+(\sum_{k_1,k_2} X_{ik_1} X_{jk_2} \Lambda_{k_1k_2}), \text{where } Z_{ij,\cdot\cdot} = 1,2,3,\ldots \tag{6}$$

Then, $\{Z_{ij,k_1k_2}\}_{k_1,k_2}$ can be obtained through the Multinomial distribution as:

$$(\{Z_{ij,k_1k_2}\}_{k_1,k_2}) \sim \text{Multinomial}\left(Z_{ij,\cdot\cdot}; \left\{\frac{X_{ik_1}X_{jk_2}\Lambda_{k_1k_2}}{\sum_{k_1,k_2} X_{ik_1}X_{jk_2}\Lambda_{k_1k_2}}\right\}_{k_1,k_2}\right) \tag{7}$$

Sampling $\{\Lambda_{k_1k_2}\}_{k_1,k_2}$ For $\Lambda_{k_1k_2}$'s posterior distribution, we get

$$P(\Lambda_{k_1k_2}|-) \propto \exp\left(-\Lambda_{k_1k_2}(\sum_{i,j} X_{ik_1}X_{jk_2})\right) \Lambda_{k_1k_2}^{\sum_{i,j} Z_{ij,k_1k_2}} \cdot \exp\left(-\Lambda_{k_1k_2}\theta_\Lambda\right)\Lambda^{k_\Lambda-1} \tag{8}$$

Thus, we get

$$\Lambda_{k_1k_2} \sim \text{Gam}\left(\sum_{i,j} Z_{ij,k_1k_2} + k_\Lambda, \frac{1}{\theta_\Lambda + \sum_{i,j} X_{ik_1}X_{jk_2}}\right) \tag{9}$$

Sampling M Given Gamma distribution $\text{Gam}(k_M, \theta_M)$ as the prior distribution for M, M's posterior distribution is:

$$P(M|-) = M^{k_M-1}\exp(-\theta_M M)\prod_{i,k}\left(\exp(-M\pi_{ik}^{(L)})\right) M^{\sum_{i,k} X_{ik}} \tag{10}$$

Thus, we sample M from:

$$M \sim \text{Gam}\left(k_M + \sum_{i,k} X_{ik}, \frac{1}{\theta_M + N}\right) \tag{11}$$

Sampling α Similarly, given Gamma distribution $\text{Gam}(k_\alpha, \theta_\alpha)$ as the prior distribution for α, α's posterior distribution is

$$\alpha \sim \text{Gam}(k_\alpha + \sum_{i,k} h_{i\alpha k}^{(1)}, \frac{1}{\theta_\alpha - \sum_i \log q_i^{(1)}}) \tag{12}$$

Computational complexity It is noted that our ssDirBN for relational modeling does not increase the computational scalability of the DirBN for relational modeling. We have changed the counts allocation in the detailed process of latent counts propagation, however, the computational complexity remained the same in our ssDirBN and DirBN. It is easy to see that the complexity of sampling the binary variable $\boldsymbol{b}$ also scales to the number of positive relations. Thus, the computational complexity of our ssDirBN for relational modeling is the same as that of DirBN.

Table 1: Dataset information. N is the number of entities, N_E is the number of positive links.

Dataset	N	N_E	Dataset	N	N_E	Dataset	N	N_E	Dataset	N	N_E
Citeer	3 312	4 715	Cora	2 708	5 429	Pubmed	2 000	17 522	PPI	4 000	105 775

5.2 Experimental results

Dataset Information We apply our ssDirBN in the following four real-world sparse datasets: three standard citation networks (*Citeer*, *Cora*, *Pubmed* [15] and one protein-to-protein interaction network (*PPI*) [23]. The number of entities N and the number of relations N_{E} for these datasets are provided in Table 1. In the citation datasets, entities correspond to documents and edges represent citation links, whereas in the protein-to-protein dataset, we model the protein-to-protein interactions [9]. We do not include the feature information of entities in the experiments. Instead, we use an identity matrix I_N as the feature matrix when these information are needed in specific models.

Evaluation Criteria We primarily focus on link prediction and use this to evaluate model performance. We use AUC (Area Under ROC Curve) and Average Precision value on test relational data as the two comparison criteria. The AUC value represents the probability that the algorithm will rank a randomly chosen existing-link higher than a randomly chosen non-existing link. Therefore, the higher the AUC value, the better the predictive performance. Each reported criteria value is the mean of 10 replicate analyses.

Experimental settings For hyper-parameters, we let $c_0 = d_0 = e_0 = f_0 = g_0 = h_0 = 0.1$ for all the datasets. The re-sampling of hyper-parameters is specified in the similar way as that of [4]. Each run uses $2\,000$ MCMC iterations with the first $1\,000$ samples discarded as burn-in and the mean value of the second $1\,000$ samples' performance score is used report the required score. Unless specified, reported AUC values are obtained using 90% (per row) of the relational data as training data and the remaining 10% as test data. After testing various scenarios for different settings of L and K, we set the number of layers $L = 3$ and the number of communities $K = 10$ for all the testing cases as the performance can be obtained in the balance of excellence performance scores and fast running time.

Comparison methods: Several Bayesian methods for relational data and two Graph Auto-Encoder models are used for comparison: the Mixed-Membership Stochastic Blockmodel [1], the Hierarchical Latent Feature Relational Model (HLFM) [10], the Node Attribute Relational Model (NARM) [19], the Hierarchical Gamma Process-Edge Partition Model (HGP-EPM) [21], graph autoencoders (GAE) and variational graph autoencoders (VGAE) [11]. The NARM, HGP-EPM, GAE and VGAE methods are executed using their respective implementations from the authors, under their default settings. The MMSB and HLFM are implemented to the best of our abilities and we set the number of layers and length of latent binary representation in HLFM as same as those in ssDirBN. For the GAE and VGAE, the AUC and Precision values are calculated based on the pairwise similarities between the entity representations.

Performance of ssDirBN for different values of K, L Fig. 3 shows the link prediction performance of ssDirBN for relational modeling on the cases of $K = 5, 10, 15, 20, 30$ and $L = 2, 3, 4, 5$. In terms of the number of communities K, we find that the performance of $K = 10$ is significantly better than the one of $K = 5$ and slightly worse than those of $K = 15, 20, 30$. The performance of $L = 3$ has similar behaviours as it is much better than that of $L = 2$, which is likely because the insufficient deep structure, and slightly worse than those of $L = 4, 5$, which may possibly due to that $L = 3$

Table 2: Link prediction performance comparison. It is noted that we do not use the entities' feature information and only use the binary relational data for each dataset.

AUC (mean and standard deviation)				
Model	Citeer	Cora	Pubmed	PPI
MMSB	0.690 ± 0.004	0.743 ± 0.007	0.774 ± 0.005	0.801 ± 0.003
NARM	0.759 ± 0.003	0.809 ± 0.003	0.808 ± 0.004	0.821 ± 0.002
HGP-EPM	0.763 ± 0.003	0.810 ± 0.003	0.803 ± 0.006	0.834 ± 0.004
HLFM	0.781 ± 0.010	0.829 ± 0.005	0.829 ± 0.005	0.856 ± 0.010
GAE	0.789 ± 0.004	0.846 ± 0.006	0.822 ± 0.004	0.874 ± 0.009
VGAE	0.790 ± 0.003	0.849 ± 0.004	0.826 ± 0.002	0.880 ± 0.007
DirBN	0.779 ± 0.004	0.832 ± 0.008	0.845 ± 0.008	0.892 ± 0.007
ssDirBN	$\mathbf{0.815} \pm 0.007$	$\mathbf{0.839} \pm 0.007$	$\mathbf{0.853} \pm 0.004$	$\mathbf{0.912} \pm 0.002$
Average Precision (mean and standard deviation)				
Model	Citeer	Cora	Pubmed	PPI
MMSB	0.661 ± 0.004	0.704 ± 0.005	0.742 ± 0.004	0.823 ± 0.003
NARM	0.781 ± 0.004	0.831 ± 0.004	0.771 ± 0.005	0.844 ± 0.002
HGP-EPM	0.776 ± 0.002	0.840 ± 0.003	0.786 ± 0.006	0.864 ± 0.004
HLFM	0.793 ± 0.004	0.842 ± 0.003	0.802 ± 0.003	0.883 ± 0.008
GAE	0.839 ± 0.004	0.884 ± 0.007	0.846 ± 0.004	0.889 ± 0.003
VGAE	0.846 ± 0.003	0.889 ± 0.004	0.850 ± 0.003	0.882 ± 0.004
DirBN	0.819 ± 0.004	0.875 ± 0.03	0.860 ± 0.007	0.884 ± 0.002
ssDirBN	$\mathbf{0.871} \pm 0.007$	$\mathbf{0.891} \pm 0.003$	$\mathbf{0.889} \pm 0.006$	$\mathbf{0.913} \pm 0.005$

is deep enough. The behaviours of AUC and Precision are quite consistent in forming these conclusions.

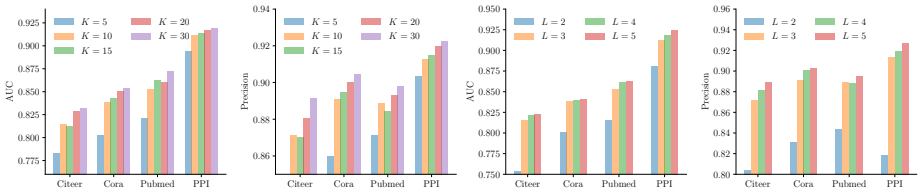


Fig. 3: Link prediction performance (AUC and Precision) of ssDirBN for relational modelling on different values of K and L.

Link Prediction performance Table 2 displays the AUC and Average Precision values over the testing relational data. As we can see, our ssDirBN performs the best among all these comparison methods. The performance of deep hierarchical models (i.e., VGAE, HLFM, DirBN, ssDirBN) are usually better than the shallow models, which verifies the

advantages of deep structures. The competitive performance of our ssDirBN, as well as the one of DirBN, over GAE and VGAE verifies the promising aspects of using Dirichlet distributions to construct the deep generative models.

Sparsity and smoothness Fig. 4 verifies that our ssDirBN can simultaneously obtain sparsity and smoothness. For sparsity, as we can see from the top row, our ssDirBN can obtain better Average Precision values and lower model complexity (larger sparsity) than the approach of DirBN. For smoothness, the bottom row shows that our ssDirBN have larger values of concentration parameters in generating the membership distributions than that of DirBN. Given the same output counts, larger concentration parameters will lead to smoother posterior distributions. Thus, the posterior distributions of $\boldsymbol{\pi}_i^{(l)}$ in our ssDirBN would be smoother than that in DirBN.

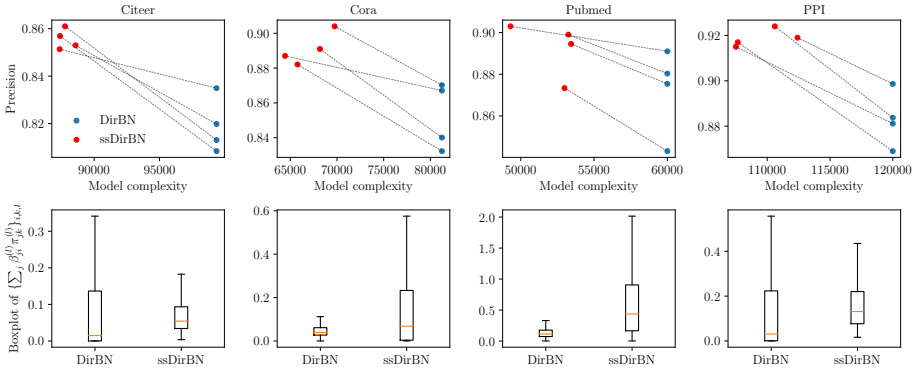


Fig. 4: Top row: model complexity versus Precision value for the datasets of *Citeer, Cora, Pubmed, PPI*. We define 'model complexity' as the number of community membership values for all entities. In DirBN, it is calculated as NKL, whereas in ssDirBN, it is calculated as: $\sum_{i,k,l} \delta_{b_{ik}^{(l)}=1}$. Each dotted line represents that its connected red and blue dots are evaluated on the same training and testing dataset. Bottom row: boxplot of concentration parameters $\{\sum_j \beta_{ji}^{(l)} \pi_{jk}^{(l)}\}_{i,k,l}$ in generating membership distributions for DirBN and ssDirBN on *Citeer, Cora, Pubmed, PPI*.

Visualizations on membership distributions Fig. 5 displays example membership distributions over the first 50 entities in the Citeer data for DirBN and ssDirBN. For DirBN, the membership distributions become more dominated by a few communities from layer 1 to layer 3. However, the components' weight ratio does not change too much. For ssDirBN, the membership distributions clearly show more changes across different layers. The representation capability of our ssDirBN can be thus enhanced through the larger changes of membership distributions across different layers.

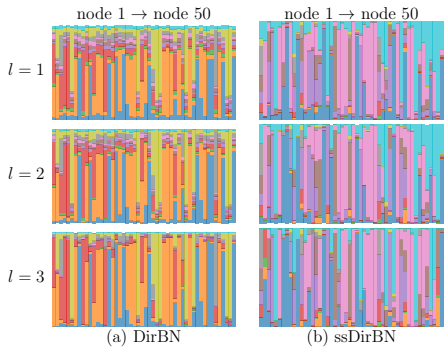


Fig. 5: Visualizations of the membership distributions ($\{\boldsymbol{\pi}_{i=1:50}^{(l)}\}_{l=1}^{3}$) on the *Citeer* data set for DirBN (left) and ssDirBN (right). The panels in the top, middle and bottom row represent the membership distributions at layer $l = 1, 2, 3$. In each panel, columns represent entities. Colors represent communities (with $K = 10$) and the length of color occupations in one column represents the membership value of the particular community for that entity.

6 Conclusion

We have decoupled the sparsity and smoothness in the Dirichlet Belief Networks (DirBN) by introducing a binary variable for each component of each entity's latent distribution at each layer. Through further model and inference modifications, we guarantee the proposed ssDirBN is well defined for the latent distributions and can be inferred by using efficient Gibbs sampling algorithm. The promising experimental results validate the effectiveness of ssDirBN over DirBN in terms of reduced model complexity and improved link prediction performance, and its competitive performance against other approaches. Given the substantial performance improvement over DirBN, we are interested in combining ssDirBN with other applications (e.g. topic modelling, collaborative filtering) in the future.

Acknowledgements

Yaqiong Li is a recipient of UTS Research Excellence Scholarship. Xuhui Fan and Scott A. Sisson are supported by the Australian Research Council (ARC) through the Australian Centre of Excellence in Mathematical and Statistical Frontiers (ACEMS, CE140100049), and Scott A. Sisson through the ARC Future Fellow Scheme (FT170100079). Bin Li is supported in part by STCSM Project (20511100400), Shanghai Municipal Science and Technology Major Projects (2018SHZDZX01, 2021SHZDZX0103), and the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning.

References

1. Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic block models. *Journal of Machine Learning Research*, 9:1981–2014, 2008.
2. Sophie Burkhardt and Stefan Kramer. Decoupling sparsity and smoothness in the dirichlet variational autoencoder topic model. *Journal of Machine Learning Research*, 20(131):1–27, 2019.
3. David B. Dunson and Amy H. Herring. Bayesian latent variable models for mixed discrete outcomes. *Biostatistics*, 6(1):11–25, 2005.
4. Xuhui Fan, Bin Li, Caoyuan Li, Scott Sisson, and Ling Chen. Scalable deep generative relational model with high-order node dependence. In *NeurIPS*, pages 12637–12647, 2019.
5. Xuhui Fan, Bin Li, Yaqiong Li, and Scott Sisson. Poisson-randomised dirbn: Large mutation is needed in dirichlet belief networks. In *ICML*, 2021.
6. Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059, 2016.
7. Dandan Guo, Bo Chen, Ruiying Lu, and Mingyuan Zhou. Recurrent hierarchical topic-guided RNN for language generation. In *ICML*, pages 3810–3821, 2020.
8. Dandan Guo, Bo Chen, Hao Zhang, and Mingyuan Zhou. Deep poisson gamma dynamical systems. In *NeurIPS*, pages 8442–8452, 2018.
9. Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.
10. Changwei Hu, Piyush Rai, and Lawrence Carin. Deep generative models for relational data with side information. In *ICML*, pages 1578–1586, 2017.
11. Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.
12. Yaqiong Li, Xuhui Fan, Ling Chen, Bin Li, Zheng Yu, and Scott A. Sisson. Recurrent dirichlet belief networks for interpretable dynamic relational data modelling. In *IJCAI*, pages 2470–2476, 2020.
13. Aaron Schein, Scott Linderman, Mingyuan Zhou, David Blei, and Hanna Wallach. Poisson-randomized gamma dynamical systems. In *NeurIPS*, pages 782–793, 2019.
14. Aaron Schein, Hanna Wallach, and Mingyuan Zhou. Poisson-gamma dynamical systems. In *NIPS*, pages 5005–5013, 2016.
15. Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. In *AI magazine*, pages 29–93, 2008.
16. Chong Wang and David Blei. Decoupling sparsity and smoothness in the discrete hierarchical dirichlet process. In *NIPS*, pages 1982–1989. Curran Associates, Inc., 2009.
17. Sikun Yang and Heinz Koeppl. A poisson gamma probabilistic model for latent node-group memberships in dynamic networks. In *AAAI*, 2018.
18. Hao Zhang, Bo Chen, Dandan Guo, and Mingyuan Zhou. WHAI: Weibull hybrid autoencoding inference for deep topic modeling. In *International Conference on Learning Representations*, 2018.
19. He Zhao, Lan Du, and Wray Buntine. Leveraging node attributes for incomplete relational data. In *ICML*, pages 4072–4081, 2017.
20. He Zhao, Lan Du, Wray Buntine, and Mingyuan Zhou. Dirichlet belief networks for topic structure learning. In *NeurIPS*, pages 7955–7966, 2018.
21. Mingyuan Zhou. Infinite edge partition models for overlapping community detection and link prediction. In *AISTATS*, pages 1135–1143, 2015.
22. Mingyuan Zhou, Yulai Cong, and Bo Chen. Augmentable gamma belief networks. *Journal of Machine Learning Research*, 17(163):1–44, 2016.
23. Marinka Zitnik and Jure Leskove. Predicting multicellular function through multi-layer tissue networks. In *Bioinformatics*, pages i190–i198, 2017.