Deep Amortized Relational Model with Group-wise Hierarchical Generative Process

Huafeng Liu[1,2], **Tong Zhou**[1], **Jiaqi Wang**[1], **Liping Jing**[1] *, **Michael K. Ng**[2]

[1]Beijing Key Lab of Traffic Data Analysis and Mining, Beijing Jiaotong University, Beijing, China
[2] Department of Mathematics, The University of Hong Kong, Hong Kong SAR, China
{huafeng, tong_zhou, jiaqi.wang, lpjing}@bjtu.edu.cn, mng@maths.hku.hk

Abstract

In this paper, we propose **D**eep **a**mortized **R**elational **M**odel (DaRM) with *group-wise hierarchical generative process* for community discovery and link prediction on relational data (e.g., graph, network). It provides an efficient neural relational model architecture by grouping nodes in a group-wise view rather than node-wise or edge-wise view. DaRM simultaneously learns what makes a group, how to divide nodes into groups, and how to adaptively control the number of groups. The dedicated group generative process is able to sufficiently exploit pair-wise or higher-order interactions between data points in both inter-group and intra-group, which is useful to sufficiently mine the hidden structure among data. A series of experiments have been conducted on both synthetic and real-world datasets. The experimental results demonstrated that DaRM can obtain high performance on both community detection and link prediction tasks.

Introduction

Relational models, which describe the pairwise interaction between nodes in a network (graph), have gained tremendous attention in recent years, with numerous methods developed to model the complex dependencies within relational data; in particular, probabilistic Bayesian methods. Aside from it usefulness in many downstream tasks, probabilistic relational model is an important tool for visualizing and understanding the underlying structure of datasets, as well as a model for categorization inn cognitive science. A plethora of relational models have been developed and successfully employed in various fields, including computer vision [Mehta, Duke, and Rai 2019], natural language processing [Larsen and Aone 1999], social network analysis [Fortunato 2010], and medical informatics [Masulli and Schenone 1999]. Among various relational models, probabilistic relational model have been widely concerned because of its flexibility and adaptivity [Zhu et al. 2017].

Probabilistic relational models [Nowicki and Snijders 2001] are a staple of statistical modeling in which a discrete latent variable is introduced for each observation, indicating its latent structures. These latent structures help discover the underlying groups in the network, as well as in predicting potential links between nodes. These generative models can be roughly divided into two categories: finite probabilistic relational model and infinite probabilistic relational model [Cohen 1982; Teh et al. 2006]. In recent years, finite probabilistic relational models have been increasingly applied in unsupervised graph learning problems with the aid of deep neural networks. The most relevant research is that of deep generative relational models [Kipf and Welling 2016; Hu, Rai, and Carin 2017; Fan et al. 2019], where neural networks are trained to predict the states of latent variables given observations in a deep generative model or probabilistic program [Sohn, Lee, and Yan 2015]. A finite relation model with a fixed number of groups may fit the given dataset well, however, it may be sub-optimal to use the same number of groups if more data comes under a slightly changed distribution. It would be ideal if probabilistic relational model can figure out the unknown number of groups.

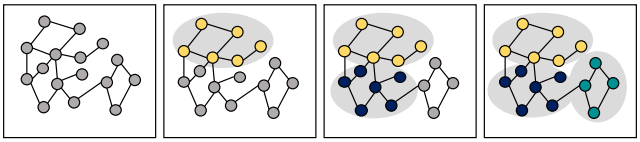


Figure 1: The Group-wise generative process of DaRM.

Alternatively, infinite probabilistic relational model is the application of nonparametric Bayesian techniques to group modeling, which allows for the automatic determination of an appropriate number of groups. The prior distribution can be specified in terms of a relation sequential process called Dirichlet process (e.g., Chinese Restaurant Process (CRP) [Kemp et al. 2006], India Buffet Process (IBP) [Miller, Jordan, and Griffiths 2009]), where the number of group can arbitrarily grow to better accommodate data as needed. To approximate the corresponding infinite relational posterior, recently, deep generative latent feature relational model [Mehta, Duke, and Rai 2019] is proposed in a black-box fashion. Specifically, it makes use of neural networks for amortized inferring the group assignments and parameters, which is flexible to define the groups. To adaptively determine the number of groups, however, they have to construct the non-parametric prior, which largely depends

*Corresponding author

on the sequential node or edge modeling. For large-scale datasets, this process will be time-consuming and difficult to converge.

In this paper, we build on these prior works and propose deep amortized relational model (DaRM) with group-wise hierarchical generative process, which aims to learn deep Bayesian posterior in a group-wise view rather than node-wise or edge-wise view, without predefining the number of groups since it allows for an unrestricted number of groups. In other words, DaRM targets on generating latent structure rather than generating nodes or relations (all existing probabilistic relational models adopted the later), as shown in Figure 1. This group-wise generative process has two good facets. One is that the whole generation process is efficient, because it depends on the number of groups (K) rather than the number of nodes (N) or edges (M), where $K \ll N$ and $K \ll M$. The other is that inter-group and intra-group structure can be sufficiently exploited during the learning process, because each group is generated according to the previous groups and the current left data points. To illustrate the superior of the proposed method, we perform experiments on both synthetic data and real-world data in terms of community detection and link prediction tasks.

Related Work

There is significant interest in probabilistic deep learning models for graphs, e.g., stochastic groups model and probabilistic relational model.

Stochastic Block Model Stochastic Block Model (SBM) is a well-studied probabilistic generative model for graph community detection with strong theoretical background [Nowicki and Snijders 2001; Karrer and Newman 2011]. A further improvement of this model introduces mixed membership by adding a Dirichlet prior to the community labels [Viroli and McLachlan 2019]. The other popular framework with strong theory is the spectral clustering[Von Luxburg 2007]. In comparison to SBM, the difficulty of the later is the costly computation of the eigenvectors and the requirement of availability of the entire adjacency matrix, and it becomes tricky for large graphs [Liu et al. 2013].

Probabilistic Relational Model There has been several work on generative models for relational data [Teh et al. 2006; Kemp et al. 2006; Konishi et al. 2015; Kipf and Welling 2016; Hu, Rai, and Carin 2017; Mehta, Duke, and Rai 2019; Fan et al. 2019]. Among them, infinite relational model (IRM) [Kemp et al. 2006], latent feature relational models (LFRM) [Miller, Jordan, and Griffiths 2009], variational bayesian infinite relational model (VBIRM) [Konishi et al. 2015] and have the ability to model infinite groups. Hu, Rai, and Carin [2017] proposed an extension of the IRM via a deep hierarchy of binary latent features for each node. However, this model relies on expensive batch MCMC inference, precluding its applicability to large-scale networks. Variational graph autoencoders (VGAE) [Kipf and Welling 2016] combines the graph neural network with variational autoencoder for modeling relational data. The recent work on deep generative model includes deep generative latent feature relational model (DGLFRM) [Mehta, Duke, and Rai 2019] and scalable deep generative relational model (SDREM)[Fan et al. 2019], which can model the relational data in more perspectives.

The Proposed Method

In this section, based on our previous work[Liu, Wang, and Jing 2021], we present the deep amortized relational model for learning latent structure from relational data in an efficient group-wise manner.

Notations and Problem Formulation

Let calligraphic letter (e.g., $\mathcal{A}$) indicate set, capital letter (e.g., A) for scalar, lower-case bold letter (e.g., $\mathbf{a}$) for vector, and capital bold letter (e.g., $\mathbf{A}$) for matrix. Relational data can usually be represented as graph or network, which is formulated as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (we will use nodes throughout this article), and $\mathcal{E}$ is the set of edges. Let $v_i \in \mathcal{V}$ to denote a node and $e_{ij} = (v_i, v_j) \in \mathcal{E}$ to denote an edge between node v_i to v_j. The relational matrix $\mathbf{A} \in \{0,1\}^{N\times N}$ is a binary matrix with $a_{i,j} = 1$ if edge $e_{i,j} \in \mathcal{E}$ and $a_{i,j} = 0$ if edge $e_{i,j} \notin \mathcal{E}$. A graph may have node attributes $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)^\top \in \mathbb{R}^{N\times D}$, where each node is represented as a D-dim vector.

Relation Modeling For groups in graphs, we assume group labels for each node are generated as in the cluster prior, followed by a generative model of relations $\mathbf{A} = \{a_{i,j}\}_{i,j=1}^N$, and possibly node features $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$. Let $\{c_1, \cdots, c_N\}$ indicate the group assignment of nodes, where $c_i \in \{1, \cdots, K\}$ denotes the group index to which the node i is assigned, and K is the number of groups. The general generative process for relational modeling is

$$c_i \sim p(c_i|\alpha_1) \quad \omega_{a,b} \sim p(\omega_{a,b}|\alpha_2) \quad a_{i,j} \sim p(\omega_{c_i,c_j}) \quad (1)$$

where α_1 and α_2 are hyperparameters, a and b indicates node group assignments and $\omega_{a,b}$ indicates latent strength between group a and b. Note that only the binary $a_{i,j}$ are observed. based on the group relationship of two nodes, the final edge between them is generated.

Group Modeling Let $\{\mathcal{C}_1, \cdots, \mathcal{C}_K\}$ indicate the final group partitions, $\mathcal{C}_k$ represents the set of nodes belonging to the k-th group. We aim to formulate the probabilistic relational model by learning the conditional joint distribution of the current group given the data points and the existing groups, $p(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathbf{X}, \mathbf{A})$, from which we can sample the groups:

$$\mathcal{C}_k \sim p(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathbf{X}, \mathbf{A}) \quad k = 1, \cdots, K \quad (2)$$

We estimate $p_\theta(\mathcal{C}_{1:K}|\mathbf{X}, \mathbf{A}) = \prod_{k=1}^K p(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathbf{X}, \mathbf{A})$ with a probabilistic deep generative model, which we describe next. Following that, it is easy to model group sequentially, and the arbitrary number of groups can be adaptively determined until there is no remaining nodes need to group.

Deep Amortized Relational Model

We propose to approximate the unknown conditional distribution $p(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathbf{X}, \mathbf{A})$, with a variational autoencoder (VAE) framework [Kingma and Welling 2013]. However, instead of directly learning a latent space for group, we model

$p(\mathcal{C}_k|\mathcal{C}_{1:k-1},\mathbf{X},\mathbf{A})$ in a hierarchical conditional distribution with a separate latent space for groups and group representations as follows:

$$p_\theta(\mathcal{C}_{1:K}|\mathbf{X},\mathbf{A}) = \prod_{k=1}^{K}\sum_{\mathcal{I}_k}\int p_\theta(\mathcal{C}_k \triangleq \mathbf{h}_k|\mathbf{z}_k,\mathcal{S}_{I_k},\mathbf{X},\mathbf{A}) \\ p_\theta(\mathbf{z}_k|\mathcal{S}_{I_k},\mathbf{X},\mathbf{A})p_\theta(\mathcal{I}_k|\mathcal{C}_{1:k-1},\mathcal{S}_k,\mathbf{X},\mathbf{A})d\mathbf{z}_k \tag{3}$$

Here the generative process $p(\mathcal{C}_k|\mathcal{C}_{1:k-1},\mathbf{X},\mathbf{A})$ is broken into three parts: 1) $p_\theta(\mathcal{I}_k|\mathcal{C}_{1:k-1},\mathcal{S}_k,\mathbf{X},\mathbf{A})$, selecting several nodes $\mathcal{I}_k$ (i.e., the indices of the nodes) from unassigned node set $\mathcal{S}_k$ for the k-th group; 2) $p_\theta(\mathbf{z}_k|\mathcal{S}_{I_k},\mathbf{X},\mathbf{A})$, generating group representation $\mathbf{z}_k \in \mathbb{R}^{D_1}$ for the k-th group; 3) $p_\theta(\mathcal{C}_k \triangleq \mathbf{h}_k|\mathbf{z}_k,\mathcal{S}_{I_k},\mathbf{X},\mathbf{A})$, making the group assignments $\mathbf{h}_k = (h_{k,1},\cdots,h_{k,M_k-|I_k|}) \in \mathbb{R}^{M_k-|I_k|}$ for the unsigned nodes except previous selected nodes $\mathcal{I}_k$. Here $|\mathcal{S}_k| = M_k$ is the number of left nodes after generating the previous $k-1$ group, defined by $M_k = N - \sum_{i=j}^{k-1} N_j$, where N_j is the number of nodes belonging to j-th group. For convenience, we denote $\{\mathcal{I}_k,\mathcal{C}_{1:k-1},\mathcal{S}_k\}$ by $\mathcal{S}_{I_k}$. The group assignments are indicated via a binary vector $\mathbf{h}_k = (h_{k,1},\cdots,h_{k,M_k-|I_k|}) \in \mathbb{R}^{M_k-|\mathcal{I}_k|}$, which can be sampled to indicate whether each left data point is affiliated with the k-th group. After obtaining $\mathbf{h}_k$, the group result $\mathcal{C}_k$ will be explicit. Obviously, it is easy to model group sequentially, and the arbitrary number of groups can be adaptively determined until there is no remaining point in $\mathcal{S}_k$.

Amortized Variational Inference Following the conditional variational auto-encoder (CVAE) paradigm [Sohn, Lee, and Yan 2015], we aim to maximize the corresponding group-wise data log-likelihood with the aid of variational distribution $q_\phi(\mathbf{z}_k,\mathcal{I}_k|\mathcal{C}_{1:k},\mathcal{S}_k,\mathbf{X},\mathbf{A})$. Model parameters θ and ϕ can be optimized by maximizing a low bound of $\mathbb{E}_{p(\mathbf{X},\mathbf{A},\mathcal{C}_{1:K})}\log p_\theta(\mathcal{C}_{1:K}|\mathbf{X},\mathbf{A})$, i.e.,

$$\mathbb{E}_{p(\mathbf{X},\mathbf{A},\mathcal{C}_{1:K})}\log p_\theta(\mathcal{C}_{1:K}|\mathbf{X},\mathbf{A}) \\ \geq -\mathbb{E}_{p(\mathbf{X},\mathbf{A},\mathcal{C}_{1:K})}\sum_{k=1}^{K}\big[KL(q_\phi(\mathbf{z}_k,\mathcal{I}_k|\mathcal{C}_{1:k},\mathcal{S}_k,\mathbf{X},\mathbf{A})|| \\ p_\theta(\mathbf{z}_k,\mathcal{I}_k|\mathcal{C}_{1:k-1},\mathbf{X},\mathbf{A})) \\ + \mathbb{E}_{q_\phi(\mathbf{z}_k,\mathcal{I}_k|\mathcal{C}_{1:k},\mathcal{S}_k,\mathbf{X},\mathbf{A})}[\log p_\theta(\mathbf{h}_k|\mathbf{z}_k,\mathcal{S}_{I_k},\mathbf{X},\mathbf{A})]\big] \tag{4}$$

Here variational distribution $q_\phi(\mathbf{z}_k,\mathcal{I}_k|\mathcal{C}_{1:k},\mathcal{S}_k,\mathbf{X},\mathbf{A})$ can be regarded as a amortized inference model which can be factorized by introducing a factorized variational posterior distribution:

$$q_\phi(\mathbf{z}_k,\mathcal{I}_k|\mathcal{C}_{1:k},\mathcal{S}_k,\mathbf{X},\mathbf{A}) = q_\phi(\mathbf{z}_k|\mathcal{S}_{I_k},\mathbf{X},\mathbf{A})q_\phi(\mathcal{I}_k|\mathcal{C}_{1:k},\mathcal{S}_k,\mathbf{X},\mathbf{A})$$

This operation will encourage group inference process from $\mathcal{I}_k$ to $\mathbf{z}_k$, i.e., $\mathbb{E}_{p(\mathbf{X},\mathbf{A},\mathcal{C}_{1:K})}[\cdot]$ and $\mathbb{E}_{q_\phi(\mathbf{z}_k,\mathcal{I}_k,|\mathcal{C}_{1:k},\mathcal{S}_k,\mathbf{X},\mathbf{A})}[\cdot]$ are intractable, and can be estimated by the Gumbel-Softmax trick [Jang, Gu, and Poole 2016] and the Gaussian re-parameterization trick [Kingma and Welling 2013], respectively. Once the training procedure is finished, the group assignment can be estimated by

$$p(\mathbf{h}_k|\mathbf{z}_k,\mathcal{C}_{1:k-1},\mathcal{S}_k,\mathbf{X},\mathbf{A}) \approx \frac{1}{R}\sum_{r=1}^{R} p_\theta(\mathbf{h}_k|\mathbf{z}_k^{(r)},\mathcal{C}_{1:k-1},\mathcal{S}_k)$$

where R is the number of Monte Carlo samples and $\mathbf{z}_k^{(r)} \sim p_\theta(\mathbf{z}_k|\mathcal{S}_{I_k},\mathbf{X},\mathbf{A})$.

Model Parameterization

In this section, we describe the detailed implementation of each components, including the node selection procedure $p_\theta(\mathcal{I}_k|\mathcal{C}_{1:k-1},\mathcal{S}_k,\mathbf{X},\mathbf{A})$, prior $p_\theta(\mathbf{z}_k|\mathcal{S}_{I_k},\mathbf{X},\mathbf{A})$, decoder $p_\theta(h_{k,i}|\mathbf{z}_k,\mathcal{S}_{I_k},\mathbf{X},\mathbf{A})$, and encoder $q_\phi(\mathbf{z}_k|\mathcal{S}_{I_k},\mathbf{X},\mathbf{A})$. Note that there are a key component are proposed to map the observed relations $\mathbf{A}$ and node features $\mathbf{X}$ together to an embedding vector space $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^N$, which is able to exploit relationship between nodes and represent data in a improved feature space. The whole model architecture of DaRM is shown in Figure 2. Furthermore, an efficient strategy is proposed to combat mode collapse. The model parameters $\{\theta,\phi\}$ include: N data point embeddings $\{\mathbf{e}_i\}_{i=1}^N \in \mathbb{R}^{N\times D}$, K pioneer data point indices $\{\mathcal{I}_k\}_{k=1}^K \in \{1,\cdots,N\}^K$ related to each group, K group prototypes $\{\mathbf{m}_k\}_{k=1}^K \in \mathbb{R}^{K\times D_1}$ and group representations $\{\mathbf{z}_k\}_{k=1}^K \in \mathbb{R}^{K\times D_1}$, K group assignments $\{\mathbf{h}_k\}_{k=1}^K$, and the parameters of neural networks. We optimize $\{\theta,\phi\}$ to maximize the training objective Eq. (4).

Graph Embedding Considering initial feature and corresponding relational information, we introduce conditional mapping module $\mathbf{E} = f(\mathbf{X},\mathbf{A})$ to model the graph embedding, which is able to map nodes in the network to a low-dimensional vector space, while saving as much structural information as possible in the representations. Graph Convolutional Network (GCN) is a class of message-passing graph neural networks that updates the representation of each node based on local neighborhood information. Inspired by previous work [Wang et al. 2020], we focus on leveraging edge gating mechanisms and each neighboring node in the graph convolution operation may receive different weights depending on the edge gate. Residual connections are used between layers for multi-layer GCN. To improve it, we define a explicitly updating edge gates across layers with information propagation:

$$\mathbf{u}_i^{l+1} = \mathbf{u}_i^l + \text{ReLU}\left(\text{BN}\left(\mathbf{W}_a^l\mathbf{u}_i^l + \sum_{j\to i} e_{ij}^l \odot \mathbf{W}_b^l\mathbf{u}_j^l\right)\right) \tag{5}$$

where $\mathbf{u}_i^l$ is the feature of node i at layer l and $\mathbf{u}_i^0 = \mathbf{x}_i$, $e_{ij}^0 = a_{ij}$. $\mathbf{W}_a^l$ and $\mathbf{W}_b^l$ are learnable weight matrix of the neural network. $\text{BN}(\cdot)$ is batch normalization. e_{ij}^l is the edge gate computed as follows:

$$e_{ij}^l = \frac{\sigma(\hat{e}_{ij}^l)}{\sum_{j'\to i}\sigma(\hat{e}_{ij'}^l)+\epsilon} \\ \hat{e}_{ij}^l = \hat{e}_{ij}^{l-1} + \text{ReLU}(\text{BN}(\mathbf{A}^l\mathbf{u}_i^{l-1} + \mathbf{B}^l\mathbf{u}_j^{l-1} + \mathbf{C}^l\hat{e}_{ij}^{l-1})) \tag{6}$$

where $\mathbf{A}^l$, $\mathbf{B}^l$ and $\mathbf{C}^l$ are learnable weight matrices. After multiple-layer process, we can obtain the embedding $\mathbf{e}_i = \mathbf{u}_i^L$ for each node (here L is the number of layers in graph embedding module).

Prototype-based Nodes Selection A straightforward approach would be to assume $p_\theta(\mathcal{I}_k|\mathcal{C}_{1:k-1},\mathcal{S}_k,\mathbf{X},\mathbf{A})$ follow a uniform distribution related to random sampling or categorical distribution with its own set of $M_k - 1$ parameters. As we all known, uniform distribution is too simple to contain any prior, and complex categorical distribution would result in over-parameterization and low sample efficiency. We instead propose a prototype-based implementation. To be specific, we introduce K group prototypes $\{\mathbf{m}_k\}_{k=1}^K$ and make

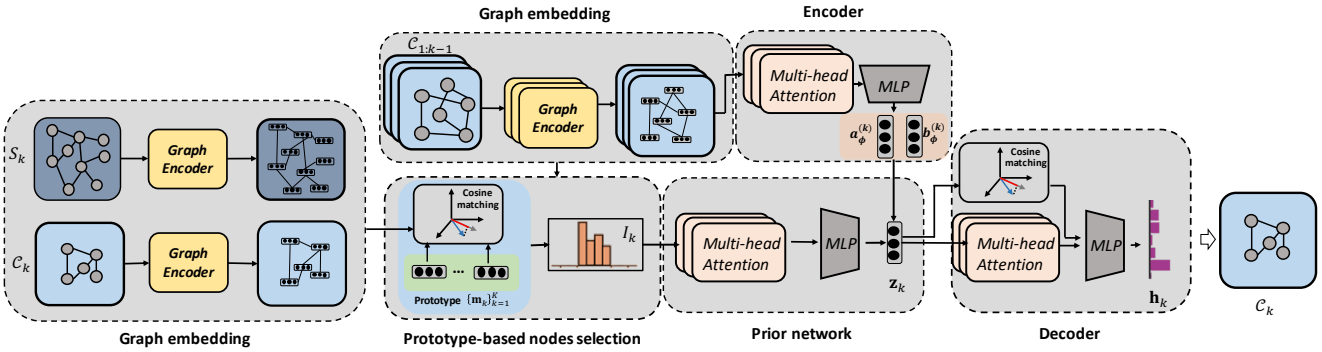


Figure 2: The model architecture of DaRM.

use of the data point representations to draw the multi-hot vector $\mathbf{o}_k$ following *Multinormal* distribution:

$$\begin{aligned} \mathbf{o}_k &\sim \textsc{Multi}\left(\textsc{Softmax}([s_1^{(k)}, \cdots, s_j^{(k)}, \cdots, s_{M_k}^{(k)}])\right) \\ s_j^{(k)} &= \frac{\textsc{Cosine}(\mathbf{e}_j^{(k)}, \mathbf{m}_k)}{\tau} \end{aligned} \tag{7}$$

Then, the indices of pioneer nodes can be obtained by $\mathcal{I}_k = \textsc{index}(\mathbf{o}_k > 0)$. $\mathbf{e}_j^{(k)}$ is the representation of the j-th data point in $\mathcal{S}_k$. Here the cosine similarity, $\textsc{Cosine}(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^\top \mathbf{b})/(||\mathbf{a}||_2||\mathbf{b}||_2)$, instead of the inner product similarity adopted by most existing deep learning methods [Luo, Schwing, and Urtasun 2016], is used to evaluate the correlation between data point and group and prevent mode collapse. In fact, with inner product, the majority of the available data points are highly like to be selected as the pioneer data point for the k-th group, which will result in improper data assignment to group. Moreover, cosine similarity can be taken as Euclidean distance on the unit hypersphere, which is more suitable for inferring the group structure than inner product [Mettes, van der Pol, and Snoek 2019]. The hyper-parameter τ scales the similarity from $[-1, 1]$ to $[-1/\tau, 1/\tau]$, which is set as $\tau = 0.1$ to obtain a more skewed distribution. The term in $s_j^{(k)}$ aims to capture the similarity between data point and the current group, which can be taken as intra-group correlation.

Prior Network The prior $p_\theta(\mathbf{z}_k|\mathcal{I}_k, \mathcal{C}_{1:k-1}, \mathcal{S}_k, \mathbf{X}, \mathbf{A}) \triangleq p_\theta(\mathbf{z}_k|\mathcal{S}_{I_k}, \mathbf{X}, \mathbf{A})$ aims to group representation for current cluster $\mathcal{C}_k$, when given the selected data points $\mathcal{I}_k$, existing groups $\mathcal{C}_{1:k-1}$ and the unsigned data points $\mathcal{S}_k$. For convenience, $\mathbf{z}_k$ is assumed following a multivariate *Gaussian* distribution with diagonal covariance matrix and sampled via $\mathcal{N}\left(\mathbf{z}_k|\boldsymbol{\mu}_\theta^{(k)}, [\text{diag}(\boldsymbol{\sigma}_\theta^{(k)})]^2\right)$ where mean $\boldsymbol{\mu}_\theta^{(k)}$ and variance $\boldsymbol{\sigma}_\theta^{(k)}$ are parameterized by neural networks $f_\theta(\cdot)$:

$$\begin{aligned} \mathbf{W}_\theta^{(k)} &= \mathbf{E}_k + f\left(\sum_{i=1}^{k-1} \textsc{Mha}(\mathbf{E}_k, \mathbf{V}_i), \textsc{Mha}(\mathbf{E}_k, \mathbf{U}_k)\right) \\ (\boldsymbol{\mu}_\theta^{(k)}, \boldsymbol{\sigma}_\theta^{(k)}) &= f_\theta(\mathbf{W}_\theta^{(k)} + f(\mathbf{W}_\theta^{(k)})) \end{aligned} \tag{8}$$

here $\mathbf{E}_k \in \mathbb{R}^{|\mathcal{I}_k| \times D}$ is the representation of $\{\mathbf{x}_i\}_{i \in \mathcal{I}_k}$, $\mathbf{V}_i \in \mathbb{R}^{N_i \times D}$ indicates node representation matrix belonging to the i-th group and $\mathbf{U}_k \in \mathbb{R}^{(M_k - |\mathcal{I}_k|) \times D}$ for the unassigned nodes. $\textsc{Mha}(\mathbf{A}, \mathbf{B})$ indicates multi-head attention for capturing the relation between $\mathbf{A}$ and $\mathbf{B}$, which is able to exploit pair-wise or higher-order interactions between data points in both inter-group and intra-group [Lee et al. 2019]. $f(\cdot)$ and f_θ are feedforward layers with layer normalization [Ba, Kiros, and Hinton 2016]. We use Multi-head attention Module $\textsc{Mha}(\cdot)$ to exploit pair-wise or higher-order interactions between data points in both inter-and intra-cluster. Considering we want to capture the elements-wise relationship between $\mathbf{A}$ and $\mathbf{B}$, we set $\mathbf{A}$ as query, and set key and values are $\mathbf{B}$. The Multi-head attention module is defined as follow:

$$\begin{aligned} \textsc{Mha}(\mathbf{A}, \mathbf{B}) &= \textsc{Concat}(\mathbf{O}_1, \cdots, \mathbf{O}_H)\mathbf{W}^h \\ \mathbf{O}_h &= \textsc{Relu}\left(\mathbf{A}\mathbf{W}_h^Q \left(\mathbf{B}\mathbf{W}_h^K\right)^\top\right)\mathbf{B}\mathbf{W}_h^V \end{aligned} \tag{9}$$

where $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V$ are head-specific transform matrices and H is the number of heads.

Decoder The decoder predicts which data points out of M_k ones are mostly to be selected to form the k-th group, i.e., $p_\theta(h_{k,i}|\mathbf{z}_k, \mathcal{S}_{I_k}) = g_{\theta,i}(\mathbf{z}_k, \mathcal{S}_{I_k})$, where we introduce neural network parameterized by $g_{\theta,i}(\cdot)$ defined in terms of

$$\begin{aligned} \mathbf{w}_i^{(k)} &= \mathbf{e}_i^{(k)} + f\left(\sum_{i=1}^{k-1} \textsc{Mha}(\mathbf{e}_i^{(k)}, \mathbf{V}_i), \textsc{Mha}(\mathbf{e}_i^{(k)}, \mathbf{U}_k)\right) \\ h_{k,i} &= g_{\theta,i}\left(\frac{\textsc{Cosine}(\mathbf{e}_i^{(k)}, \mathbf{z}_k)}{\tau}, \mathbf{w}_i^{(k)} + f\left(\mathbf{w}_i^{(k)}\right)\right) \end{aligned} \tag{10}$$

Encoder The inference procedure contains two parts: $q_\phi(\mathcal{I}_k|\mathcal{C}_{1:k}, \mathcal{S}_k, \mathbf{X}, \mathbf{A})$ and $q_\phi(\mathbf{z}_k|\mathbf{h}_k, \mathcal{S}_{I_k}, \mathbf{X}, \mathbf{A})$. For $\mathcal{I}_k$, the variational distribution has the same architecture as generative distribution. Due to the page limitation, we omit it and the full description can be found in supplementary material. Similarly, $\mathbf{z}_k$ is sampled via a multivariable *Gaussian* distribution, i.e., $q_\phi(\mathbf{z}_k|\mathbf{h}_k, \mathcal{S}_{I_k}, \mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{z}_k|\boldsymbol{\mu}_\phi^{(k)}, [\text{diag}(\boldsymbol{\sigma}_\phi^{(k)})]^2)$, here mean and standard deviation are parameterized by a neural network $f_\phi(\cdot)$:

$$\begin{aligned} (\mathbf{a}_\phi^{(k)}, \mathbf{b}_\phi^{(k)}) &= f_\phi\left(\frac{\sum_{j=1}^{N_k} h_{k,j} \cdot \mathbf{e}_j^{(k)}}{\sqrt{\sum_{j=1}^{N_k}(h_{k,j})^2}}, \sum_{i=1}^{k} \textsc{Mha}(\mathbf{E}_k, \mathbf{V}_i)\right) \\ \boldsymbol{\mu}_\phi^{(k)} &= \frac{\mathbf{a}_\phi^{(k)}}{||\mathbf{a}_\phi^{(k)}||_2} \qquad \boldsymbol{\sigma}_\phi^{(k)} \leftarrow \sigma_0 \cdot \exp\left(-\frac{1}{2}\mathbf{b}_\phi^{(k)}\right) \end{aligned} \tag{11}$$

where $h_{k,j} = 1$ if $j \in \mathcal{I}_k$. The neural network $f_\phi(\cdot)$ captures nonlinearity, and is shared across the K components.

Table 1: Dataset statistics.

Dataset	*NIPS12*	*Cora*	*CiteSeer*	*Pubmed*
♯ nodes (N)	2,037	2,708	3,312	19,717
♯ relations (M)	3,134	5,429	4,715	44,324
♯ features (D)	100	1,433	3,703	500
♯ groups (K)	12	7	6	3

We normalize the mean to be consistent with the use of cosine similarity which projects the representations onto a unit hypersphere. Note that σ_0 should be set to a small value, e.g., around 0.1, since the learned representations are well normalized.

Experiments

We report experimental results on both synthetic and real-world datasets in terms of community detection performance and link prediction performance.

Experimental Setup

Dataset Several widely known citation datasets are used, namely, *NIPS12* [Globerson et al. 2007], *Cora*, *CiteSeer* and *Pubmed* [Rossi and Ahmed 2015]. More detailed information of datasets is given in Table 1.

Baselines We compare the proposed model with two kinds of methods, including traditional methods: infinite relational model (IRM) [Kemp et al. 2006], variational bayesian infinite relational model (VBIRM) [Konishi et al. 2015], latent feature relational models (LFRM) [Miller, Jordan, and Griffiths 2009]; Deep generative models: variational autoencoder on graphs (VGAE) [Kipf and Welling 2016], hierarchical latent feature model (HLFM) [Hu, Rai, and Carin 2017], deep generative latent feature relational model (DGLFRM) [Mehta, Duke, and Rai 2019], scalable deep generative relational model (SDREM)[Fan et al. 2019].

Metrics For community detection task, two evaluation metrics are adopted. One is internal evaluation, *Davies-Bouldin Index (DBI)* [Davies and Bouldin 1979]: $DBI = \frac{1}{K}\sum_{k=1}^{K}\max_{j\neq k}\left((a_k + a_j)/d(\mathbf{c}_k, \mathbf{c}_j)\right)$ where $\mathbf{c}_k$ is the centroid of the k-th group, a_k is the average distance of all elements in group k to centroid $\mathbf{c}_k$. $d(\cdot)$ is cosine similarity. Smaller DBI value indicates better performance. The other is external evaluation including group accuracy (*ACC*) and Normalized Mutual Information (*MNI*). *ACC* is defined by $ACC = \max_{m\in\mathcal{M}}\frac{1}{N}\mathbb{1}\{y_i = m(\hat{y}(\mathbf{x}_i))\}$, where y_i is the ground-truth label that corresponds to that $\mathbf{x}_i$ sample, $\hat{y}(\mathbf{x}_i)$ is the group assignment obtained by the model, and m ranges over the set $\mathcal{M}$ of all possible one-to-one mappings between group assignments and labels. Normalized Mutual Information $NMI(\mathcal{C}_i, \mathcal{C}_j) = \frac{MI(\mathcal{C}_i,\mathcal{C}_j)}{\sqrt{H(\mathcal{C}_i)H(\mathcal{C}_j)}}$, where N is the total number of data samples, y_i is the ground-truth label that corresponds to that $\mathbf{x}_i$ sample, $\hat{y}(\mathbf{x}_i)$ is the cluster assignment obtained by the model, and m ranges over the set $\mathcal{M}$ of all possible one-to-one mappings between cluster assignments and labels. Larger *ACC* value indicates better performance. For link prediction performance evaluation on relational data, we use *AUC* (Area Under ROC Curve) as the comparison criteria. The *AUC* value represents the probability that the method 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.

Experiments on Synthetic Data

We first demonstrate DaRM on synthetic graph data with arbitrary number of groups. We generate dataset by the following process.

$$N \sim \mathrm{Unif}(0.3N_{max}, N_{max}) \qquad (c_i)_{i=1}^{N} \sim \mathrm{CRP}(\alpha)$$
$$(\lambda_{c_i,c_j})_{i,j=1}^{N} \sim N(0, \sigma_\lambda^2) \qquad (a_{i,j})_{i,j=1}^{N} \sim \mathrm{Bernoulli}(\lambda_{c_i,c_j})$$

where N_{max} is set to 500 and CRP is a Chinese Restaurant Process with concentration parameter $\alpha = 0.7$.

We consider four baselines as comparison: IRM, VBIRM, DGLFRM and SDREM. Among them, IRM and VBIRM are traditional relational model with different inference methods, and DGLFRM and SDREM are deep neural network based methods. Two testing scenarios (*s1* and *s2*) are constructed to evaluate the effectiveness of the proposed method. The testing set in *s1* has the same configuration (200 samples and 4 groups) as training set, while *s2* contains different numbers of samples and groups (400 samples and 6 groups) in order to verify whether the proposed method can generalize to the unseen groups. As shown in Table 2, we list community detection and link prediction performance on two testing scenarios in terms of *DBI*, *ACC*, *NMI* and *AUC*. As expected, the proposed DaRM consistently outperforms baselines on both s_1 and s_2. Although *s2* is more challenging, DaRM can capture group uncertainty and obtain the best results. In model efficiency evaluation, DaRM obtains the fast running time. The main reason is the group-wise strategy is much more efficient than relation-wise strategy.

Experiments on Real-world Data

We conduct experiments on the real-world data to evaluate the proposed method on community detection and link prediction performance. Table 3 summarizes the results on four datasets in terms of *DBI*, *ACC* and *NMI*. It can be seen that our DaRM outperforms all baselines in terms of both intern and external evaluation metrics. The main reason, we believe, is that DaRM not only exploits amortized properties between data points in both inter-group and intra-group, but also has flexible generative process, which simultaneously emphasizes the superiority of variational generative framework. For link prediction task, we hold out 10% and 5% of the links as our test set and validation set, respectively, and use the validation set to fine-tune the hyperparameters. We take the average of *AUC* scores by running model on 10 random split of dataset. As shown in Table 3. Our model, DaRM, outperforms the baselines on almost all datasets. We again highlight that unlike the baselines, such as VGAE, DGLFRM and SDREM, that directly model the connections among nodes, our model learns embeddings and intrinsic latent structure together. In terms of running time, thanks to the group-wise sequential modeling strategy, DaRM achieves the best results in the shortest time.

Table 2: Performance on synthetic relational data

	Metric	IRM	VBIRM	DGLFRM	SDREM	DaRM
s1	*DBI*	$0.0313_{\pm 0.005}$	$0.0426_{\pm 0.001}$	$0.0367_{\pm 0.002}$	$0.0301_{\pm 0.002}$	$\mathbf{0.0218}_{\pm 0.003}$
	ACC	$0.9566_{\pm 0.021}$	$0.9432_{\pm 0.003}$	$0.9623_{\pm 0.004}$	$0.9615_{\pm 0.003}$	$\mathbf{0.9698}_{\pm 0.002}$
	NMI	$0.8977_{\pm 0.002}$	$0.8934_{\pm 0.005}$	$0.9345_{\pm 0.004}$	$0.9389_{\pm 0.003}$	$\mathbf{0.9573}_{\pm 0.004}$
	AUC	$0.8344_{\pm 0.001}$	$0.8521_{\pm 0.002}$	$0.8663_{\pm 0.003}$	$0.8715_{\pm 0.004}$	$\mathbf{0.8857}_{\pm 0.003}$
	Time[s]	$0.0821_{\pm 0.032}$	$0.0585_{\pm 0.021}$	$0.0314_{\pm 0.047}$	$0.0421_{\pm 0.056}$	$\mathbf{0.0179}_{\pm 0.061}$
s2	*DBI*	$0.0554_{\pm 0.014}$	$0.0827_{\pm 0.006}$	$0.0761_{\pm 0.003}$	$0.0587_{\pm 0.004}$	$\mathbf{0.0512}_{\pm 0.004}$
	ACC	$0.9331_{\pm 0.007}$	$0.9244_{\pm 0.005}$	$0.9421_{\pm 0.003}$	$0.9414_{\pm 0.003}$	$\mathbf{0.9426}_{\pm 0.004}$
	NMI	$0.8532_{\pm 0.004}$	$0.8644_{\pm 0.015}$	$0.8934_{\pm 0.006}$	$0.8873_{\pm 0.004}$	$\mathbf{0.9236}_{\pm 0.003}$
	AUC	$0.8455_{\pm 0.006}$	$0.8677_{\pm 0.003}$	$0.8843_{\pm 0.004}$	$0.8802_{\pm 0.002}$	$\mathbf{0.9056}_{\pm 0.003}$
	Time[s]	$0.0831_{\pm 0.032}$	$0.0525_{\pm 0.015}$	$0.0335_{\pm 0.045}$	$0.0483_{\pm 0.032}$	$\mathbf{0.0157}_{\pm 0.061}$

Table 3: Comparing clustering performance (*DBI*, *ACC* and *NMI*), link prediction performance (*AUC*) and running time on four real-world datasets.

Dataset	Metric	LFRM	HLFM	VGAE	DGLFRM	SDREM	DaRM
NIPS12	*DBI*	$0.4354_{\pm 0.001}$	$1.5946_{\pm 0.002}$	$0.2394_{\pm 0.001}$	$0.2578_{\pm 0.001}$	$0.2327_{\pm 0.002}$	$\mathbf{0.2142}_{\pm 0.001}$
	ACC	$0.9446_{\pm 0.002}$	$0.8630_{\pm 0.002}$	$0.9758_{\pm 0.002}$	$0.9596_{\pm 0.003}$	$0.9796_{\pm 0.002}$	$\mathbf{0.9832}_{\pm 0.001}$
	NMI	$0.6832_{\pm 0.005}$	$0.7531_{\pm 0.004}$	$0.7498_{\pm 0.004}$	$0.7454_{\pm 0.003}$	$0.7642_{\pm 0.002}$	$\mathbf{0.7867}_{\pm 0.002}$
	AUC	$0.8489_{\pm 0.003}$	$0.8733_{\pm 0.005}$	$0.8790_{\pm 0.004}$	$0.8894_{\pm 0.002}$	$0.8769_{\pm 0.003}$	$\mathbf{0.8963}_{\pm 0.002}$
	Time[s]	$23.44_{\pm 2.24}$	$27.83_{\pm 1.32}$	$21.42_{\pm 1.02}$	$24.65_{\pm 0.78}$	$26.33_{\pm 2.16}$	$\mathbf{13.62}_{\pm 1.27}$
Cora	*DBI*	$0.7606_{\pm 0.004}$	$0.7435_{\pm 0.003}$	$0.6958_{\pm 0.004}$	$0.6828_{\pm 0.004}$	$0.6845_{\pm 0.002}$	$\mathbf{0.6542}_{\pm 0.001}$
	ACC	$0.6245_{\pm 0.003}$	$0.6362_{\pm 0.004}$	$0.6631_{\pm 0.004}$	$0.7138_{\pm 0.005}$	$0.7123_{\pm 0.003}$	$\mathbf{0.7342}_{\pm 0.002}$
	NMI	$0.4883_{\pm 0.004}$	$0.5024_{\pm 0.003}$	$0.5342_{\pm 0.002}$	$0.5551_{\pm 0.003}$	$0.5531_{\pm 0.003}$	$\mathbf{0.5732}_{\pm 0.002}$
	AUC	$0.9096_{\pm 0.003}$	$0.9127_{\pm 0.004}$	$0.9260_{\pm 0.002}$	$0.9277_{\pm 0.003}$	$0.9271_{\pm 0.001}$	$\mathbf{0.9365}_{\pm 0.002}$
	Time[s]	$15.34_{\pm 1.11}$	$13.22_{\pm 2.17}$	$14.52_{\pm 1.17}$	$21.52_{\pm 1.19}$	$20.41_{\pm 2.25}$	$\mathbf{9.32}_{\pm 1.34}$
CiteSeer	*DBI*	$0.5730_{\pm 0.031}$	$0.5246_{\pm 0.042}$	$0.5313_{\pm 0.045}$	$0.4863_{\pm 0.032}$	$0.4734_{\pm 0.027}$	$\mathbf{0.4566}_{\pm 0.015}$
	ACC	$0.6384_{\pm 0.001}$	$0.6596_{\pm 0.003}$	$0.6630_{\pm 0.004}$	$0.6733_{\pm 0.003}$	$0.6869_{\pm 0.002}$	$\mathbf{0.6932}_{\pm 0.003}$
	NMI	$0.3632_{\pm 0.004}$	$0.3873_{\pm 0.003}$	$0.3983_{\pm 0.002}$	$0.4032_{\pm 0.004}$	$0.4355_{\pm 0.003}$	$\mathbf{0.4452}_{\pm 0.002}$
	AUC	$0.8965_{\pm 0.002}$	$0.9154_{\pm 0.001}$	$0.9001_{\pm 0.003}$	$0.9146_{\pm 0.004}$	$0.9033_{\pm 0.002}$	$\mathbf{0.9233}_{\pm 0.002}$
	Time[s]	$36.55_{\pm 2.57}$	$42.31_{\pm 1.19}$	$33.21_{\pm 2.11}$	$36.59_{\pm 2.01}$	$34.31_{\pm 1.05}$	$\mathbf{14.46}_{\pm 1.25}$
Pubmed	*DBI*	$0.9921_{\pm 0.044}$	$0.9822_{\pm 0.027}$	$0.9502_{\pm 0.036}$	$0.8923_{\pm 0.035}$	$0.8906_{\pm 0.043}$	$\mathbf{0.8733}_{\pm 0.032}$
	ACC	$0.6446_{\pm 0.002}$	$0.6553_{\pm 0.001}$	$0.6704_{\pm 0.003}$	$0.6852_{\pm 0.002}$	$0.6741_{\pm 0.003}$	$\mathbf{0.7032}_{\pm 0.002}$
	NMI	$0.2877_{\pm 0.003}$	$0.3043_{\pm 0.004}$	$0.3001_{\pm 0.003}$	$0.3122_{\pm 0.003}$	$0.3233_{\pm 0.002}$	$\mathbf{0.3431}_{\pm 0.002}$
	AUC	$0.9152_{\pm 0.003}$	$0.9214_{\pm 0.004}$	$0.9318_{\pm 0.002}$	$0.9416_{\pm 0.002}$	$0.9122_{\pm 0.001}$	$\mathbf{0.9514}_{\pm 0.001}$
	Time[s]	$351.21_{\pm 8.72}$	$435.11_{\pm 6.15}$	$354.11_{\pm 6.77}$	$387.12_{\pm 8.77}$	$351.52_{\pm 7.62}$	$\mathbf{152.33}_{\pm 6.74}$

Analysis of Model Behavior

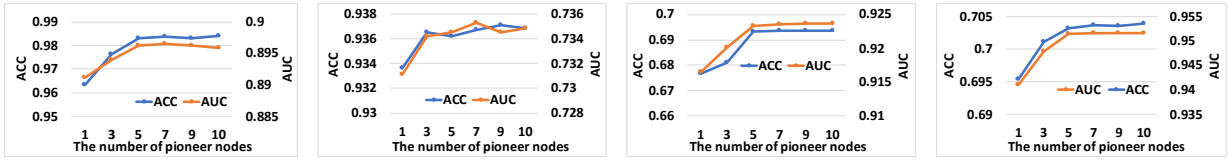
To further understand model behavior, we conduct several experiments to investigate the effect of parameters and list some detailed experiments results.

Selected Pioneer Nodes To form the k-th group, DaRM tends to select some pioneer nodes $\mathcal{I}_k$. Actually, the number of selected pioneer nodes plays an important role in forming a more accurate group. In order to investigate the effect of pioneer nodes, we conduct experiments by setting different number of pioneer nodes. Figure 3 shows the effect of different number of pioneer nodes in terms of *ACC* and *AUC*. We can see that DaRM obtains the best results (around 5 for *NIPS12*, *CiteSeer* and *Pubmed*, and 3 for *Cora*), after that, the performance remained basically stable. This result is in line with our intuition that the more the numbers, the better the performance and as the number increases, the increase of performance decreases. One possible reason is that a smaller number of pioneer nodes can provide enough guidance for the follow-up process.

Different Settings of L The number of layers L in graph embedding module is a crucial parameter to control the model capacity on learning improved node representations. Figure. 4 shows the effect of L on all nine datasets in terms of clustering performance and running time. It can be seen that the overall accuracy is improved with the increasing of layers. After the performance reaches the best value, with the increase of the number of layers, the performance tends to be stable, while the computational complexity is still increasing.

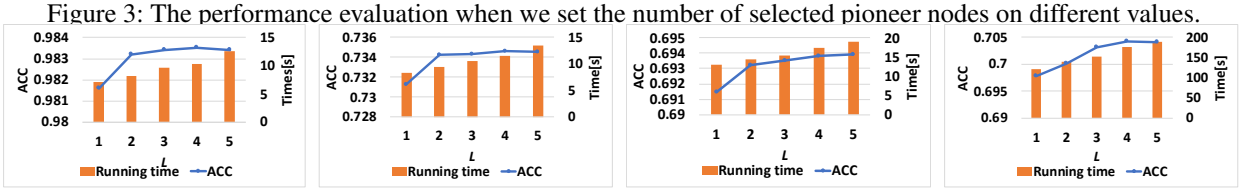
Cosine Similarity vs. Inner Product In our model, the cosine similarity, instead of inner product similarity, is used to calculate the correlations between each pair of data point and prototype. In this experiment, a new model with inner product similarity is trained to compare that with DaRM with cosine similarity. Table 4 lists the performance comparison between DaRM with inner product similarity (DaRM-inner) and DaRM with cosine similarity (DaRM-cos). We can see that DaRM-cos is superior to DaRM-inner. This result confirms that selecting a proper metric, such as cosine similarity, is important for model training.

Groups Interpretation Taking *NIPS12* as an example, we demonstrate top-5 authors obtained by DaRM in three representative groups including 'probabilistic modeling', 'neural



(a) *NIPS12* (b) *Cora* (c) *CiteSeer* (d) *Pubmed*

Figure 3: The performance evaluation when we set the number of selected pioneer nodes on different values.



(a) *NIPS12* (b) *Cora* (c) *CiteSeer* (d) *Pubmed*

Figure 4: The effect of L, the number of layers in the graph embedding module in terms of running time (seconds) and accuracy (*ACC*).

Table 4: Comparison between DaRM with inner product similarity and DaRM with cosine similarity.

Metric	Method	*NIPS12*	*Cora*	*CiteSeer*	*Pubmed*
DBI	DaRM-inner	$0.0289_{\pm 0.002}$	$0.6811_{\pm 0.003}$	$0.4766_{\pm 0.029}$	$0.8915_{\pm 0.025}$
	DaRM-cos	$\mathbf{0.0218}_{\pm 0.003}$	$\mathbf{0.6542}_{\pm 0.001}$	$\mathbf{0.4566}_{\pm 0.015}$	$\mathbf{0.8733}_{\pm 0.032}$
ACC	DaRM-inner	$0.9636_{\pm 0.001}$	$0.7142_{\pm 0.003}$	$0.6884_{\pm 0.002}$	$0.6757_{\pm 0.003}$
	DaRM-cos	$\mathbf{0.9832}_{\pm 0.001}$	$\mathbf{0.7342}_{\pm 0.002}$	$\mathbf{0.6932}_{\pm 0.003}$	$\mathbf{0.7032}_{\pm 0.002}$
NMI	DaRM-inner	$0.7658_{\pm 0.003}$	$0.5564_{\pm 0.001}$	$0.4376_{\pm 0.003}$	$0.3253_{\pm 0.002}$
	DaRM-cos	$\mathbf{0.7867}_{\pm 0.001}$	$\mathbf{0.5732}_{\pm 0.002}$	$\mathbf{0.4452}_{\pm 0.002}$	$\mathbf{0.3431}_{\pm 0.002}$
AUC	DaRM-inner	$0.8659_{\pm 0.002}$	$0.9254_{\pm 0.001}$	$0.6834_{\pm 0.002}$	$0.6789_{\pm 0.003}$
	DaRM-cos	$\mathbf{0.8857}_{\pm 0.003}$	$\mathbf{0.9365}_{\pm 0.002}$	$\mathbf{0.9233}_{\pm 0.002}$	$\mathbf{0.9414}_{\pm 0.001}$

Table 5: Selected top-5 pioneer nodes by DaRM on *NIPS12*.

Group	Selected top-5 pioneer nodes (authors)
Probabilistic Modeling	Sejnowski T, Jordan M, Hinton G, Frey B J, Ghahramani Z
Neural Networks	LeCun Y, Sejnowski T, Hinton G, Benjio Y, Tang A
Reinforcement Learning	Connolly C, Michel A, Eshr K, Peper F, Thrun S

networks' and 'reinforcement learning', as shown in Table 5. Each of group represent a sub-filed, with authors working on similar topics. It can be seen that these selected nodes have unique characteristics of their corresponding groups, which will further leverage the group generative process.

Conclusion and Future Work

In this paper, we proposed Deep amortized Relational Model (DaRM) for modeling latent group structure on relational data in a group-wise strategy. We showed how the proposed model can be effectively applied in the community detection and link prediction problem, and we achieve competitive to better performance with significant save of time. It will be interesting to investigate the model other application with relational data (e.g., collaborative filtering) in the future.

Acknowledgments

This work was partly supported by the Beijing Natural Science Foundation under Grant Z180006; The National Natural Science Foundation of China under Grant 62176020, 61822601, and 61632004; The National Key Research and Development Program (2020AAA0106800); The Fundamental Research Funds for the Central Universities (2019JBZ110); CAAI-Huawei MindSpore Open Fund; the Open Project Program Foundation of the Key Laboratory of Opto-Electronics Information Processing, Chinese Academy of Sciences(OEIP-O-202004). The research of Michale K. Ng is partially supported by the Research Grants Council of Hong Kong, General Research Fund (12300218, 12300519, 17201020 and 17300021). We are also grateful for the anonymous reviewers and the editor for their helpful comments.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Cohen, R. B. A. C. 1982. Finite mixture distributions. *Technometrics*, 24(4): 339.

Davies, D. L.; and Bouldin, D. W. 1979. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2): 224–227.

Fan, X.; Li, B.; Li, C.; SIsson, S.; and Chen, L. 2019. Scalable deep generative relational model with high-order node dependence. In *Proceedings of the NeurIPS*, 12658–12668.

Fortunato, S. 2010. Community detection in graphs. *Physics reports*, 486(3-5): 75–174.

Globerson, A.; Chechik, G.; Pereira, F.; and Tishby, N. 2007. Euclidean embedding of co-occurrence data. *The Journal of Machine Learning Research*, 8: 2265–2295.

Hu, C.; Rai, P.; and Carin, L. 2017. Deep generative models for relational data with side information. In *Proceedings of the ICML*, 1578–1586.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Karrer, B.; and Newman, M. E. 2011. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1): 016107.

Kemp, C.; Tenenbaum, J. B.; Griffiths, T. L.; Yamada, T.; and Ueda, N. 2006. Learning systems of concepts with an infinite relational model. In *Proceedings of the AAAI*, volume 3, 5.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Konishi, T.; Kubo, T.; Watanabe, K.; and Ikeda, K. 2015. Variational bayesian inference algorithms for infinite relational model of network data. *IEEE Transactions on Neural Networks and Learning Systems*, 26(9): 2176–2181.

Larsen, B.; and Aone, C. 1999. Fast and effective text mining using linear-time document clustering. In *Proceedings of the ACM SIGKDD*, 16–22.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.; Choi, S.; and Teh, Y. W. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the ICML*, 3744–3753. PMLR.

Liu, H.; Wang, J.; and Jing, L. 2021. Cluster-wise hierarchical generative model for deep amortized clustering. In *Proceedings of the CVPR*, 15109–15118.

Liu, J.; Wang, C.; Danilevsky, M.; and Han, J. 2013. Large-scale spectral clustering on graphs. In *Proceedings of the IJCAI*.

Luo, W.; Schwing, A. G.; and Urtasun, R. 2016. Efficient deep learning for stereo matching. In *Proceedings of the CVPR*.

Masulli, F.; and Schenone, A. 1999. A fuzzy clustering based segmentation system as support to diagnosis in medical imaging. *Artificial intelligence in medicine*, 16(2): 129–147.

Mehta, N.; Duke, L. C.; and Rai, P. 2019. Stochastic Blockmodels meet Graph Neural Networks. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the ICML*, volume 97, 4466–4474.

Mettes, P.; van der Pol, E.; and Snoek, C. 2019. Hyperspherical prototype networks. In *Proceedings of the NeurIPS*, 1487–1497.

Miller, K.; Jordan, M.; and Griffiths, T. 2009. Nonparametric latent feature models for link prediction. In *Proceedings of the NeurPS*, volume 22, 1276–1284.

Nowicki, K.; and Snijders, T. A. B. 2001. Estimation and prediction for stochastic blockstructures. *Journal of the American statistical association*, 96(455): 1077–1087.

Rossi, R. A.; and Ahmed, N. K. 2015. The network data repository with interactive graph analytics and visualization. In *AAAI*.

Sohn, K.; Lee, H.; and Yan, X. 2015. Learning structured output representation using deep conditional generative models. In *Proceedings of the NeurIPS*, 3483–3491.

Teh, Y. W.; Jordan, M. I.; Beal, M. J.; and Blei, D. M. 2006. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476): 1566–1581.

Viroli, C.; and McLachlan, G. J. 2019. Deep gaussian mixture models. *Statistics and Computing*, 29(1): 43–51.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416.

Wang, Y.; Lee, Y.; Basu, P.; Lee, J.; Teh, Y. W.; Paninski, L.; and Pakman, A. 2020. Amortized Probabilistic Detection of Communities in Graphs. *arXiv preprint arXiv:2010.15727*.

Zhu, J.; Chen, J.; Hu, W.; and Zhang, B. 2017. Big learning with Bayesian methods. *National Science Review*, 4(4): 627–651.