Co-embedding Attributed Networks

Zaiqiao Meng*
Sun Yat-sen University
King Abdullah University of Science and Technology
zqmeng@aliyun.com

Shangsong Liang
Sun Yat-sen University
King Abdullah University of Science and Technology
liangshangsong@gmail.com

Hongyan Bao
King Abdullah University of Science and Technology
hybao.ld@gmail.com

Xiangliang Zhang†
King Abdullah University of Science and Technology
xiangliang.zhang@kaust.edu.sa

ABSTRACT

Existing embedding methods for attributed networks aim at learning low-dimensional vector representations for nodes only but not for both nodes and attributes, resulting in the fact that they cannot capture the affinities between nodes and attributes. However, capturing such affinities is of great importance to the success of many real-world attributed network applications, such as attribute inference and user profiling. Accordingly, in this paper, we introduce a Co-embedding model for Attributed Networks (CAN), which learns low-dimensional representations of both attributes and nodes in the same semantic space such that the affinities between them can be effectively captured and measured. To obtain high-quality embeddings, we propose a variational auto-encoder that embeds each node and attribute with means and variances of Gaussian distributions. Experimental results on real-world networks demonstrate that our model yields excellent performance in a number of applications compared with state-of-the-art techniques.

KEYWORDS

Attributed Network; Network Embedding; Variational Auto-encoder

ACM Reference Format:
Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding Attributed Networks. In *Proceedings of The 12th ACM International WSDM Conference (WSDM'2019).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Network embedding techniques that learn low-dimensional vector representations of nodes in a network have been attracting increasing attention in recent years. Performance of various network-based applications, such as node classification [29, 33], node clustering [3], community detection [10, 38], and link prediction [23], has been shown to be significantly improved via utilizing embeddings of nodes inferred by network embedding techniques.

Attributed networks as one category of the most important networks are ubiquitous in a myriad of important domains, ranging from social media networks to academic networks, where rich attributes, e.g., ages of the users and journals/conferences the authors published at, describing the properties of the nodes are available. A number of attributed network embedding algorithms have been proposed to learn low-dimensional vector representations of the nodes via leveraging the structure and/or the attribute information of the networks [11, 13, 13, 21, 24, 39]. By doing so, embeddings of the nodes in attributed networks can be effectively obtained, which in turn help to enhance the performance of many down-stream applications [11, 13, 20].

However, these embedding algorithms suffer from a number of defects: (1) They represent each node by a single point in a low-dimensional continuous space, resulting in the fact that the **uncertainty** of nodes' representations is ignored. Yet uncertainty is inherent when describing a node by a single point only [2]; (2) They focus on learning the representations for nodes but not for **both nodes and attributes**, resulting in the fact that the affinities between nodes and attributes are impossible to be effectively measured. However, obtaining such affinities between nodes and attributes is of great importance to the success of many tasks, where the relationships between nodes and attributes need to be quantitatively measured, such as connecting users and keywords in user profiling [25], annotating users with tags in attribute inference [9, 40] and characterizing experts by topics in expert finding [15].

To alleviate the aforementioned problems, we introduce a **C**o-embedding model for **A**ttribute **N**etworks, abbreviated as CAN, which aims at learning low-dimensional vector representations of both attributes and nodes in a same semantic space such that the affinities between them can be effectively captured and measured. To effectively infer the embeddings of both nodes and attributes in the network, we propose a variational auto-encoder, where we infer the embeddings of both nodes and attributes and represent them by means of Gaussian distributions, with the corresponding variances measuring the uncertainty of the inferred embeddings.

The node and attribute embeddings obtained in the unified manner in CAN can benefit not only **node-oriented** network problems (e.g., node classification and link prediction), but also **attribute inference** problems (e.g., predicting the value of attributes of nodes). More importantly, the common semantic embedding space provides a simple but effective solution to **user profiling** problem, as the relevance of users (nodes) and keywords (attributes) can be directly measured, e.g., by cosine similarity, or dot product. To verify the effectiveness of our proposed CAN model, we conduct experiments on seven real-world network datasets. Our experimental results demonstrate that CAN yields excellent performance in attribute inference and user profiling tasks that the previous network embedding models cannot deal with, while it also shows high competitive performance in link prediction and node classification tasks compared with state-of-the-art techniques.

*Zaiqiao Meng was at King Abdullah University of Science and Technology when this work was carried out.
†Xiangliang Zhang is the corresponding author.

Our contributions in this work can be summarized as follows:

(1) We propose a novel co-embedding algorithm for attributed networks, CAN, which aims at learning the low-dimensional representations of both nodes and attributes in the same semantic space, such that the affinities between nodes and attributes of the networks can be effectively measured.
(2) To learn high-quality embeddings of both nodes and attributes in attributed networks, we propose a variational auto-encoder method, which consists of an inference model, aiming at encoding features of nodes and attributes into two Gaussian distributions, and a generative model, which can effectively reconstructs both real and binary weighted edges and attributes.
(3) We conduct extensive experiments on real-world attributed networks to verify the effectiveness of our embedding model on addressing four graph mining tasks, i.e., node classification, link prediction, attribute inference and user profiling. Experimental results show that our model is able to learn informative and high-quality representations and significantly outperforms the state-of-the-art methods.

2 RELATED WORK

There are two lines of work related to our model, network embedding and variational auto-encoders.

2.1 Network Embedding

Existing network embedding algorithms can be classified into two categories: those [3, 10, 29, 33, 36] for plain networks where only topological structure information is utilized for embedding and those [11, 13, 21, 24, 39] for non-plain networks such as attributed networks where not only topological structure information but also auxiliary information, e.g., content information/attributes of the nodes, are taken into account for embedding.

To embed plain networks, approaches such as DeepWalk [29], node2vec [10] and LINE [33] learn embeddings based on random walks or edge sampling. These methods utilize the Skip-Gram with negative sampling neural network architecture originally proposed for word embedding [27], and can be implicitly approximated by matrix factorization [30]. Wang et al. [36] proposed the SDNE model to capture the highly non-linear network structure and preserve the global and local structure of the network. Wang et al. [38] incorporate the community structure of network into result embeddings that preserves both of the microscopic and community structures.

Some other works obtain embeddings for non-plain networks with rich auxiliary information, such as labels, node attributes and text contents, in addition to the topological structure networks [13, 37, 39]. Huang et al. [13] employ the graph Laplacian technique to learn the joint embedding from the topological structure and attributes. Kipf and Welling [21] propose a graph convolutional neural network model for attributed networks, and further employ it on a variational auto-encoder architecture [20]. Hamilton et al. [11] propose the GraphSAGE model that learns node representations by sampling and aggregating features from a node's local neighborhood. Zhang et al. [41] and Gao and Huang [8] propose their customized deep neural network architectures, called ANRL and DANE respectively, to learn node embeddings while capturing the underlying high non-linearity in both topological structure and attributes. Their results showed that combining different types of auxiliary information, rather than using only the topological features, can provide different insights of embedding of nodes, and helps to capture rich patterns in many real-world networks. However, these models only obtain embeddings of nodes and do not capture the uncertainty inherent in the embeddings.

Recently, few approaches embed nodes by distributions and capture the uncertainty of the embeddings [2, 5, 12]. Inspired by previous work of learning Gaussian word embeddings [35], the KG2E model [12] represents each entity/relation of knowledge graphs as a Gaussian distribution. Dos Santos et al. [5] study heterogeneous graphs for node classification using Gaussian embeddings. In terms of embedding for attributed networks, Bojchevski and GÃijnnemann [2] embed each node as a Gaussian distribution according to the energy-based loss of personalized ranking formulation. We tackle the embedding problem for attributed networks by modeling both nodes and attributes as Gaussian distributions, which allows measure the uncertainty of both embeddings by the variances of the distributions.

2.2 Variational Auto-encoders

Variational Auto-encoders (VAEs) are expressive latent variable models that can be used to learn complex probability distributions from training data. A basic VAE model [18, 19, 32] is comprised a pipeline of two computational neural networks: one acts as an inference network encoding the given observations into latent variables, and the other acts as a generative model performing the reverse mapping from latent variables to each observed data point. By using the variational evidence lower bound of the log-likelihood as loss function and the reparameterization tricks [19], these models can be easily trained via a Stochastic Gradient Variational Bayes estimator [19]. Many variations of VAE have been proposed [16, 18, 20, 28, 32], which have been extensively studied and applied in various tasks such as semi-supervised classification [18], clustering [16, 26] and image generation [6]. The GAE [20] model proposed by Kipf and Welling is an example of VAEs that learns representations for attributed networks. However, it learns embeddings for nodes only, rather than both nodes and attributes.

3 NOTATIONS AND THE PROBLEM

In this section, we introduce the used notations and formally define the studied problem.

3.1 Notations

In this paper, scalars are denoted by normal alphabets (e.g., the total number of nodes in the network: N), while sets are denoted by calligraphy typeface alphabets (e.g., the set of nodes in the network: $\mathcal{V}$). Matrices are denoted by boldface uppercase alphabets (e.g., an adjacency matrix $\mathbf{A}$ serving for the network). The ith row of a matrix, e.g., $\mathbf{A}$, is denoted with a subscript, e.g., $\mathbf{A}_i$. Vectors are represented by boldface lowercase alphabets or uppercase alphabets with a subscript (e.g., $\mathbf{a}$ or $\mathbf{A}_i$). A vector with a subscript represents a scalar element of that vector (e.g., $\mathbf{a}_i$ or $\mathbf{A}_{ij}$). The transpose of a matrix, e.g. $\mathbf{A}$, is represented by $\mathbf{A}^{\mathrm{T}}$.

Definition 1. *Let $\mathcal{V}$ and $\mathcal{A}$ be a set of nodes and attributes, respectively, in a graph/network. An **Attributed Network (AN)** is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{E}^{\mathcal{V}}, \mathcal{E}^{\mathcal{A}})$ with $N = |\mathcal{V}|$ nodes and*

Table 1: Main notations in our paper.

Symbol	Description
$\mathcal{G}$	an undirected attributed network
$\mathcal{V}$	set of nodes
$\mathcal{A}$	set of attributes
$\mathcal{E}^{\mathcal{V}}$	set of edges
$\mathcal{E}^{\mathcal{A}}$	set of node-attribute associations
$N = \lvert\mathcal{V}\rvert$	number of nodes
$F = \lvert\mathcal{A}\rvert$	number of attributes
D	dimension of latent variables
$\mathbf{A} \in \mathbb{R}^{N\times N}$	adjacency matrix of nodes
$\mathbf{X} \in \mathbb{R}^{N\times F}$	attribute information matrix for nodes
$\mathbf{Z}^{\mathcal{V}} \in \mathbb{R}^{N\times D}$	latent representation matrix for all the *nodes*
$\mathbf{Z}^{\mathcal{A}} \in \mathbb{R}^{F\times D}$	latent representation matrix for all the *attributes*

$F = |\mathcal{A}|$ attributes, where $\mathcal{E}^{\mathcal{V}} \subseteq \mathcal{V}\times\mathcal{V}$ is the set of edges and $\mathcal{E}^{\mathcal{A}} \subseteq \mathcal{V}\times\mathcal{A}$ is the set of node-attribute associations.

We introduce an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ and an attribute matrix of nodes $\mathbf{X} \in \mathbb{R}^{N\times F}$ for $\mathcal{G}$, with $\mathbf{A}_i$ being the i-th row vector of $\mathbf{A}$ and $\mathbf{X}_i$ being the i-th row vector of $\mathbf{X}$. The scalar elements $\mathbf{A}_{ij} \in \mathbf{A}_i$ and $\mathbf{X}_{ij} \in \mathbf{X}_i$ are weights of edges and attributes respectively, which can be either real or binary values. $\mathbf{A}$ and $\mathbf{X}$ can be regarded as the features of nodes, so we can also use $\mathbf{X}^{\mathbf{T}}$ to represent the input feature of attributes, i.e., the nodes affiliated to each attribute are regarded as its features. Tab. 1 summarizes our main notations.

3.2 Problem Definition

With the terminologies described above, we formally define the problem of co-embedding an attribute network as follows:

Problem 1. ***Attributed Network Co-embedding.*** *Given an attribute graph $\mathcal{G}$, the goal is to learn a mapping function Ξ that satisfies the following in an unsupervised way:*

$$\mathcal{G} \xrightarrow{\Xi} \mathbf{Z}^{\mathcal{V}}, \mathbf{Z}^{\mathcal{A}}, \tag{1}$$

such that both structure and attribute information of the network can be preserved as much as possible by $\mathbf{Z}^{\mathcal{V}} \in \mathbb{R}^{N\times D}$ and $\mathbf{Z}^{\mathcal{A}} \in \mathbb{R}^{F\times D}$, respectively, with $\mathbf{Z}_i^{\mathcal{V}} \in \mathbf{Z}^{\mathcal{V}}$, $\mathbf{Z}_j^{\mathcal{A}} \in \mathbf{Z}^{\mathcal{A}}$, N, F and D being the embeddings of the i-th node and the j-th attribute to be inferred, the number of nodes and attributes, and the size of the dimension of the embeddings, respectively.

4 CO-EMBEDDING NODES AND ATTRIBUTES

To address the aboved-defined problem, we introduce CAN, a model for **C**o-embedding **A**ttributed **N**etworks based on the Variational Auto-Encoder [19], to obtain the Gaussian embeddings of both nodes and attributes. Fig. 1 provides an overview of our model. We now describe the model in the following subsections.

4.1 Variational Lower Bound

To obtain embeddings of both nodes and attributes for an attributed network $\mathcal{G}$ in an unsupervised way, i.e., $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$, we first define an objective to maximize the log-likelihood of the observed adjacency matrix $\mathbf{A}$ and attribute matrix $\mathbf{X}$ of $\mathcal{G}$. Using Jensen's inequality, the log-likelihood of $\mathbf{A}$ and $\mathbf{X}$ for a given attributed network $\mathcal{G}$ can be represented as:

$$\begin{aligned}
\log p(\mathbf{A},\mathbf{X}) &= \log \int_{\mathbf{Z}^{\mathcal{V}}}\int_{\mathbf{Z}^{\mathcal{A}}} p(\mathbf{A},\mathbf{X},\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}})\mathrm{d}\mathbf{Z}^{\mathcal{V}}\mathrm{d}\mathbf{Z}^{\mathcal{A}} \\
&= \log \int_{\mathbf{Z}^{\mathcal{V}}}\int_{\mathbf{Z}^{\mathcal{A}}} p(\mathbf{A},\mathbf{X},\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}})\frac{q_\phi(\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}}\mid\mathbf{A},\mathbf{X})}{q_\phi(\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}}\mid\mathbf{A},\mathbf{X})}\mathrm{d}\mathbf{Z}^{\mathcal{V}}\mathrm{d}\mathbf{Z}^{\mathcal{A}} \\
&\geq \mathbb{E}_{q_\phi(\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}}\mid\mathbf{A},\mathbf{X})}\left[\log\frac{p(\mathbf{A},\mathbf{X},\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}})}{q_\phi(\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}}\mid\mathbf{A},\mathbf{X})}\right],
\end{aligned} \tag{2}$$

where $q_\phi(\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}}\mid\mathbf{A},\mathbf{X})$, abbreviated as q_ϕ for convenient discussion, is the variational posterior to approximate the true posterior $p(\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}}\mid\mathbf{A},\mathbf{X})$ with ϕ being the parameters needed to be estimated. Here we assume q_ϕ to be a mean-field distribution and can be factorized as:

$$q_\phi(\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}}\mid\mathbf{A},\mathbf{X}) = \prod_{i\in\mathcal{V}} q_{\phi_1}(\mathbf{Z}_i^{\mathcal{V}}\mid\mathbf{A},\mathbf{X})\prod_{a\in\mathcal{A}} q_{\phi_2}(\mathbf{Z}_a^{\mathcal{A}}\mid\mathbf{X}^{\mathbf{T}}). \tag{3}$$

The joint distribution $p_\theta(\mathbf{A},\mathbf{X},\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}})$ can be represented as:

$$\begin{aligned}
p_\theta(\mathbf{A},\mathbf{X},\mathbf{Z}^{\mathcal{V}},\mathbf{Z}^{\mathcal{A}}) = p(\mathbf{Z}^{\mathcal{V}})p(\mathbf{Z}^{\mathcal{A}})&\prod_{(i,j)\in\mathcal{E}_{\mathcal{V}}} p_{\theta_1}(\mathbf{A}_{ij}\mid\mathbf{Z}_i^{\mathcal{V}},\mathbf{Z}_j^{\mathcal{V}}) \\
&\prod_{(i,a)\in\mathcal{E}_{\mathcal{A}}} p_{\theta_2}(\mathbf{X}_{ia}\mid\mathbf{Z}_i^{\mathcal{V}},\mathbf{Z}_a^{\mathcal{A}}),
\end{aligned} \tag{4}$$

where $\mathbf{A}_{ij} \in \mathbf{A}$, $\mathbf{X}_{ia} \in \mathbf{X}$ are weights of edges and attributes respectively. Substituting Eq. 3 and Eq. 4 into Eq. 2, Eq. 2 can be represented as:

$$\begin{aligned}
\log p(\mathbf{A},\mathbf{X}) \geq\ & \mathbb{E}_{q_\phi}\left[\sum_{(i,j)\in\mathcal{E}_{\mathcal{V}}}\log p_\theta(\mathbf{A}_{ij}\mid\mathbf{Z}_i^{\mathcal{V}},\mathbf{Z}_j^{\mathcal{V}})\right] \\
&+ \mathbb{E}_{q_\phi}\left[\sum_{(i,a)\in\mathcal{E}_{\mathcal{A}}}\log p_\theta(\mathbf{X}_{ia}\mid\mathbf{Z}_i^{\mathcal{V}},\mathbf{Z}_a^{\mathcal{A}})\right] \\
&- D_{KL}(q_\phi(\mathbf{Z}^{\mathcal{V}}\mid\mathbf{A},\mathbf{X})\parallel p(\mathbf{Z}^{\mathcal{V}})) \\
&- D_{KL}(q_\phi(\mathbf{Z}^{\mathcal{A}}\mid\mathbf{X}^{\mathbf{T}})\parallel p(\mathbf{Z}^{\mathcal{A}})) \\
\triangleq\ & \mathcal{L}(\theta,\phi;\mathbf{A},\mathbf{X}),
\end{aligned} \tag{5}$$

where $\mathcal{L}(\theta,\phi;\mathbf{A},\mathbf{X})$ is the evidence lower bound (ELBO) on the marginal likelihood of the observed variables and $D_{KL}(\cdot\parallel\cdot)$ is the Kullback-Leibler (KL) divergence. In Eq. 5, $q_{\phi_1}(\mathbf{Z}^{\mathcal{V}}\mid\mathbf{A},\mathbf{X})$ and $q_{\phi_2}(\mathbf{Z}^{\mathcal{A}}\mid\mathbf{X}^{\mathbf{T}})$ are referred to the *probabilistic encoders*, since given the observed data, i.e., $\mathbf{A}$ and $\mathbf{X}$, they aim at producing variational posterior distributions over the possible values of the latent embeddings $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$. Similarly, we refer to $p_{\theta_1}(\mathbf{A}_{ij}\mid\mathbf{Z}_i^{\mathcal{V}},\mathbf{Z}_j^{\mathcal{V}})$ and $p_{\theta_2}(\mathbf{X}_{ia}\mid\mathbf{Z}_i^{\mathcal{V}},\mathbf{Z}_a^{\mathcal{A}})$ as the *probabilistic decoders*, since given the latent embeddings $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$, they produce a distribution over the possible corresponding values of observed edges and attributes. The KL-divergence terms in Eq. 5 can be interpreted as the regularizer encouraging the approximate posteriors to be close to the priors $p(\mathbf{Z}^{\mathcal{V}})$ and $p(\mathbf{Z}^{\mathcal{A}})$, while the expectation terms are regarded as expected negative reconstruction error loss.

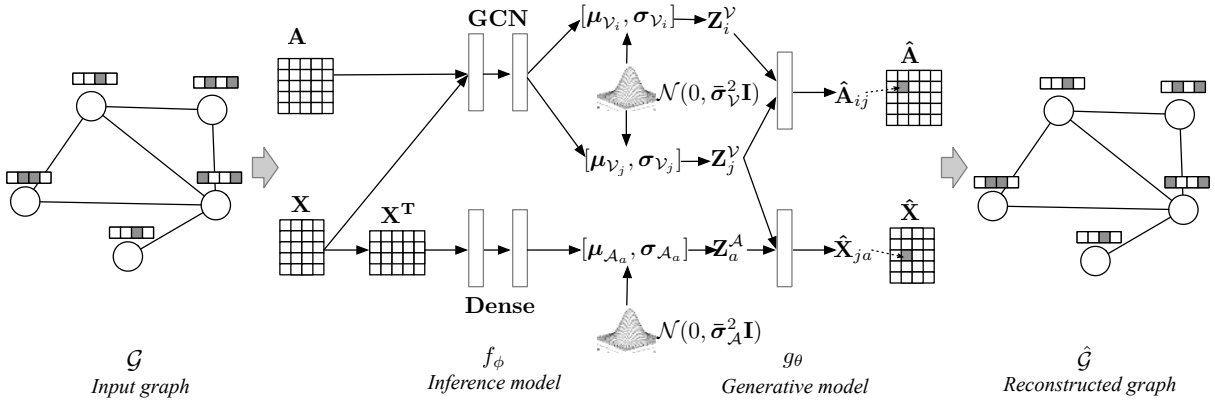

Figure 1: The architecture of our proposed CAN model. The model takes the adjacency matrix A and the attribute matrix X as input and outputs Gaussian distributions with means and variances as latent embeddings for all nodes and attributes of the network, respectively. The two neural network models, i.e., the inference model f_ϕ and the generative model g_θ, act as the probabilistic encoder and the probabilistic decoder respectively.

Commonly, analytical solutions of expectations w.r.t. the variational posterior are intractable in general cases, but we can reduce the problem of estimating the gradient w.r.t. parameters of the posterior distribution to a simpler problem of estimating the gradient w.r.t. parameters of a deterministic function, which is called the *reparameterization* trick [19].

Similar to VAE [19], we assume all of the priors of latent variables and the variational posterior distributions to be Gaussian:

$$p(\mathbf{Z}_i^{\mathcal{V}}) = \mathcal{N}(\mathbf{0}, \bar{\boldsymbol{\sigma}}_{\mathcal{V}}^2 \mathbf{I}), \tag{6}$$

$$p(\mathbf{Z}_a^{\mathcal{A}}) = \mathcal{N}(\mathbf{0}, \bar{\boldsymbol{\sigma}}_{\mathcal{A}}^2 \mathbf{I}), \tag{7}$$

$$q_{\phi_1}(\mathbf{Z}_i^{\mathcal{V}} \mid \mathbf{A}_i, \mathbf{X}_i) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{V}_i}, \boldsymbol{\sigma}_{\mathcal{V}_i}^2 \mathbf{I}), \tag{8}$$

$$q_{\phi_2}(\mathbf{Z}_a^{\mathcal{A}} \mid \mathbf{X}_a^{\mathrm{T}}) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{A}_a}, \boldsymbol{\sigma}_{\mathcal{A}_a}^2 \mathbf{I}), \tag{9}$$

where $\bar{\boldsymbol{\sigma}}_{\mathcal{V}}^2$, $\bar{\boldsymbol{\sigma}}_{\mathcal{A}}^2$ are hyperparameters of the priors, $\boldsymbol{\mu}_{\mathcal{V}_i}$ and $\boldsymbol{\mu}_{\mathcal{A}_a}$ $\boldsymbol{\sigma}_{\mathcal{V}_i}^2$ and $\boldsymbol{\sigma}_{\mathcal{A}_a}^2$ are means and variances of node embeddings and attributed embeddings to be learned, respectively.

Since we assume the priors and the variational posteriors are Gaussian distributions, the KL-divergence terms in Eq. 5 have analytical forms. By using the stochastic gradient variational Bayes (SGVB) estimator and the *reparameterization* trick [19], we can directly derivate from Monte Carlo estimates of these expectation terms by the following estimators of this model:

$$\mathcal{L}(\theta, \phi; \mathbf{A}, \mathbf{X}) = \frac{1}{N^2 L} \sum_{l=1}^{L} \Big(\sum_{i,j \in \mathcal{V}} \log p_{\theta_1}(\mathbf{A}_{ij} \mid \mathbf{Z}_i^{\mathcal{V}(l)}, \mathbf{Z}_j^{\mathcal{V}(l)}) \Big)$$

$$+ \frac{1}{NFL} \sum_{l=1}^{L} \Big(\sum_{i \in \mathcal{V}, a \in \mathcal{A}} \log p_{\theta_2}(\mathbf{X}_{ia} \mid \mathbf{Z}_i^{\mathcal{V}(l)}, \mathbf{Z}_a^{\mathcal{A}(l)}) \Big)$$

$$+ \frac{3}{2N} \sum_{i \in \mathcal{V}} \sum_{d=1}^{D} \Big(1 + \log(\frac{\boldsymbol{\sigma}_{\mathcal{V}_i}^2 \mid_d}{\bar{\boldsymbol{\sigma}}_{\mathcal{V}_i}^2 \mid_d}) - \frac{(\boldsymbol{\mu}_{\mathcal{V}_i} \mid_d)^2}{\bar{\boldsymbol{\sigma}}_{\mathcal{V}_i}^2 \mid_d} - \boldsymbol{\sigma}_{\mathcal{V}_i}^2 \mid_d \cdot \bar{\boldsymbol{\sigma}}_{\mathcal{V}_i}^2 \mid_d \Big)$$

$$+ \frac{1}{2F} \sum_{a \in \mathcal{A}} \sum_{d=1}^{D} \Big(1 + \log(\frac{\boldsymbol{\sigma}_{\mathcal{A}_a}^2 \mid_d}{\bar{\boldsymbol{\sigma}}_{\mathcal{A}_a}^2 \mid_d}) - \frac{(\boldsymbol{\mu}_{\mathcal{A}_a} \mid_d)^2}{\bar{\boldsymbol{\sigma}}_{\mathcal{A}_a}^2 \mid_d} - \boldsymbol{\sigma}_{\mathcal{A}_a}^2 \mid_d \cdot \bar{\boldsymbol{\sigma}}_{\mathcal{A}_a}^2 \mid_d \Big),$$

where

$$\begin{aligned} \mathbf{Z}_i^{\mathcal{V}(l)} &= \boldsymbol{\mu}_{\mathcal{V}_i} + \boldsymbol{\sigma}_{\mathcal{V}_i}^2 \odot \boldsymbol{\epsilon}^{(l)}, \text{with } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}), \\ \mathbf{Z}_j^{\mathcal{V}(l)} &= \boldsymbol{\mu}_{\mathcal{V}_j} + \boldsymbol{\sigma}_{\mathcal{V}_j}^2 \odot \boldsymbol{\epsilon}^{(l)}, \text{with } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}), \\ \mathbf{Z}_a^{\mathcal{A}(l)} &= \boldsymbol{\mu}_{\mathcal{A}_a} + \boldsymbol{\sigma}_{\mathcal{A}_a}^2 \odot \boldsymbol{\epsilon}^{(l)}, \text{with } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}), \end{aligned} \tag{10}$$

where L is size of the sampling, $\bullet \mid_d$ denotes the d-th element of $\bullet$, D is the size of dimension of the latent vectors, and $\odot$ indicates the element-wise product, $\boldsymbol{\epsilon}$ is the auxiliary noise variable, and $\mathcal{N}(0, \mathbf{I})$ is the standard normal distribution. As shown in Eq. 10, by a differentiable transformation of an auxiliary 'noise' variable $\boldsymbol{\epsilon}^{(l)}$ [19], we can reparameterize the latent Gaussian embeddings $q_{\phi_1}(\mathbf{Z}^{\mathcal{V}} \mid \mathbf{A}, \mathbf{X})$ and $q_{\phi_2}(\mathbf{Z}^{\mathcal{A}} \mid \mathbf{X}^{\mathrm{T}})$ into $\mathbf{Z}^{\mathcal{V}(l)}$ and $\mathbf{Z}^{\mathcal{A}(l)}$, which are deterministic and can be differentiated efficiently using the backpropagation algorithm. Hence, the vanilla gradient-based optimization techniques can be used to train the model by end-to-end.

4.2 Optimization

To optimize the objective in Eq. 10, we apply two neural network models, i.e., the *inference model* f_ϕ with trainable parameters ϕ and the *generative model* g_θ with trainable parameters θ, for the probabilistic encoder and probabilistic decoder, respectively, to perform gradient ascent for learning all the model parameters.

Inference model f_ϕ. To encode from observation variables to Gaussian embeddings, we apply a two-layer GCN [21] and a two-layer fully connected neural networks composed of non-linear mapping functions to map the adjacency matrix $\mathbf{A}$ and attribute matrix $\mathbf{X}$ of $\mathcal{G}$ to the means and variations of the variational posterior distributions (i.e. Gaussian embeddings) of nodes and attributes. We apply GCN but not other neural network models for nodes' inference, as it is able to yield the best performance. In particular,

the two-layer GCN is defined as:

$$\mathbf{H}^{(1)}_{\mathcal{V}} = \text{ReLU}\left(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}_{\mathcal{V}}\right),$$
$$\left[\boldsymbol{\mu}_{\mathcal{V}}, \boldsymbol{\sigma}^2_{\mathcal{V}}\right] = \tilde{\mathbf{A}}\mathbf{H}^{(1)}_{\mathcal{V}}\mathbf{W}^{(1)}_{\mathcal{V}}, \tag{11}$$

where $\boldsymbol{\mu}_{\mathcal{V}}$ and $\boldsymbol{\sigma}^2_{\mathcal{V}}$ are the means and variances of the learned Gaussian embeddings of nodes (see Eq. 8), $\text{ReLU}(\cdot) = max(0, \cdot)$ is the non-linear activation function, $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ being $\mathcal{G}$'s degree matrix, and $\phi_1 = [\mathbf{W}^{(0)}_{\mathcal{V}}, \mathbf{W}^{(1)}_{\mathcal{V}}]$ are trainable weights for the node inference layers, respectively.

The two fully connected layers for inferring Gaussian embeddings of attributes are defined as:

$$\mathbf{H}^{(1)}_{\mathcal{A}} = \tanh\left(\mathbf{X}^{\mathrm{T}}\mathbf{W}^{(0)}_{\mathcal{A}} + \mathbf{b}^{(0)}\right),$$
$$\left[\boldsymbol{\mu}_{\mathcal{A}}, \boldsymbol{\sigma}^2_{\mathcal{A}}\right] = \mathbf{H}^{(1)}_{\mathcal{A}}\mathbf{W}^{(1)}_{\mathcal{A}} + \mathbf{b}^{(1)}, \tag{12}$$

where $\boldsymbol{\mu}_{\mathcal{A}}$ and $\boldsymbol{\sigma}^2_{\mathcal{A}}$ are the means and variances of the learned embeddings of attributes (see Eq. 9), $\mathbf{b}$ is the bias, $\tanh(\cdot)$ is the tangent activation function and $\phi_2 = [\mathbf{W}^{(0)}_{\mathcal{A}}, \mathbf{W}^{(1)}_{\mathcal{A}}, \mathbf{b}^{(0)}, \mathbf{b}^{(1)}]$ are trainable weights for the attribute inference layers. Let $\phi = [\phi_1, \phi_2]$ denote all the trainable parameters in inference model.

After having obtained all the means and variances for all the Gaussian embeddings of nodes and attributes, the reparameterization trick [19] is applied to transform from the latent Gaussian random variables to the deterministic variables $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$, which are differentiable and capable to propagation the gradient between the inference model and generative model.

Generative model g_θ. The generative model is to decode from deterministic latent embeddings $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$ to generative random variables, where the original input edges and attributes can be reconstructed. Specifically, given embeddings of two nodes i and j, we compute $\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{V}}(i,j)}$ and $\boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{V}}(i,j)}$ by:

$$[\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{V}}(i,j)}, \boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{V}}(i,j)}] = g_{\theta_1}(\mathbf{Z}^{\mathcal{V}}_i, \mathbf{Z}^{\mathcal{V}}_j), \tag{13}$$

where g_{θ_1} is a neural network for reconstructing edges. Then an observed edge can be generated by the following:

(a) For real-valued edges,

$$p_{\theta_1}(\mathbf{A}_{ij} \mid \mathbf{Z}^{\mathcal{V}}_i, \mathbf{Z}^{\mathcal{V}}_j) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{V}}(i,j)}, \boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{V}}(i,j)}\mathbf{I}). \tag{14}$$

(b) For binary edges,

$$p_{\theta_1}(\mathbf{A}_{ij} \mid \mathbf{Z}^{\mathcal{V}}_i, \mathbf{Z}^{\mathcal{V}}_j) = \text{Ber}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{V}}(i,j)}). \tag{15}$$

where $\mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{V}}(i,j)}, \boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{V}}(i,j)}\mathbf{I})$ and $\text{Ber}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{V}}(i,j)})$ are multivariate Gaussian distribution and Bernoulli distribution parametrized by $\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{V}}(i,j)}$, $\boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{V}}(i,j)}\mathbf{I}$ and $\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{V}}(i,j)}$, respectively.

Similarly, given embeddings of node i and attribute a, we compute $\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{A}}(i,a)}$ and $\boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{A}}(i,a)}$ by:

$$[\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{A}}(i,a)}, \boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{A}}(i,a)}] = g_{\theta_2}(\mathbf{Z}^{\mathcal{V}}_i, \mathbf{Z}^{\mathcal{A}}_a), \tag{16}$$

where g_{θ_2} is a neural network for reconstructing attributes. Then an observed attribute association $(i, a) \in \mathcal{E}_{\mathcal{A}}$ can be generated by the following process:

(a) For real-valued attributes,

$$p_{\theta_2}(\mathbf{X}_{ia} \mid \mathbf{Z}^{\mathcal{V}}_i, \mathbf{Z}^{\mathcal{A}}_a) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{A}}(i,a)}, \boldsymbol{\sigma}^2_{\mathcal{E}_{\mathcal{A}}(i,a)}\mathbf{I}). \tag{17}$$

(b) For binary attributes,

$$p_{\theta_2}(\mathbf{X}_{ia} \mid \mathbf{Z}^{\mathcal{V}}_i, \mathbf{Z}^{\mathcal{A}}_a) = \text{Ber}(\boldsymbol{\mu}_{\mathcal{E}_{\mathcal{A}}(i,a)}). \tag{18}$$

We let $\theta = [\theta_1, \theta_2]$ denote the packed trainable parameters in the inference model. As all the edges and attributes in our experimental datasets are binary-valued, we implement model g simply by inner product between latent variables, i.e.,

$$g_{\theta_1}(\mathbf{Z}^{\mathcal{V}}_i, \mathbf{Z}^{\mathcal{V}}_j) = \text{sigmoid}(\mathbf{Z}^{\mathcal{V}\mathrm{T}}_i\mathbf{Z}^{\mathcal{V}}_j),$$
$$g_{\theta_2}(\mathbf{Z}^{\mathcal{V}}_i, \mathbf{Z}^{\mathcal{A}}_a) = \text{sigmoid}(\mathbf{Z}^{\mathcal{V}\mathrm{T}}_i\mathbf{Z}^{\mathcal{A}}_a), \tag{19}$$

where $\text{sigmoid}(\cdot)$ is the sigmoid function. Our experimental results show that such a simple implementation produces better results than state-of-the-art baselines in many graph mining tasks such as link prediction, node classification and attribute inference.

5 EXPERIMENTAL SETUP

In this section, we detail our experimental setup, i.e., the research questions (§5.1), the used datasets (§5.2), and the to compare and experimental settings (§5.3).

5.1 Research Questions

We aim at answering the following research questions for evaluating our proposed CAN model:

(**RQ**1) How does our CAN perform by utilizing the learned representations in traditional graph mining tasks, e.g., link prediction and node classification?
(**RQ**2) Can our CAN perform better than other models in tasks of attribute inference and user profiling, where capturing the affinities between nodes and attributes is crucial?
(**RQ**3) Do the obtained embeddings of nodes and attributes have high-quality and can we qualitatively evaluate them?

5.2 Datasets

We conduct experiments on the following attributed network datasets, the statistics of which is provided in Tab. 2:

- **Cora, Citeseer, Pubmed** [31]: The Cora, Citeseer and Pubmed datasets are citation networks, where nodes are publications and edges are citation links. Attributes of each node are bag-of-words representations of the corresponding publications.
- **BlogCatalog** [34]: This is a network of social relationships of bloggers from the BlogCatalog website, where nodes' attributes are constructed by keywords, which are generated by users as a short description of their blogs. The labels represent the topic categories provided by the authors.
- **Flickr** [14]: It is a social network where nodes represent users and edges correspond to friendships among users. The labels represent the interest groups of the users.
- **Facebook** [22]: This network is built from profile and relation data of 10 users in Facebook by SNAP[1]. The attributes are constructed by their profiles.

[1] Available from:http://snap.stanford.edu/data/.

Table 2: Statistics of datasets.

Datasets	#Nodes	#Edges	#Attributes	#Labels
Cora	2,708	5,429	1,433	7
Citeseer	3,312	4,660	3,703	6
Pubmed	19,717	44,338	500	3
BlogCatalog	5,196	171,743	8,189	6
Flickr	7,575	239,738	12,047	9
Facebook	4,039	88,234	1,406	-
DBLP	12,213	131,713	172	-

- **DBLP**: This dataset is crawled from the DBLP public bibliography data [2]. We build a coauthor network extracted from the publications in top 172 computer science conferences (Tiers A and B from China Computer Federation [3]). We treat each author as a node, each collaboration between two authors in a publication as an edge. The 172 conferences are viewed as the attributes of each node.

5.3 Baselines and Settings

We compare CAN with two categories of methods: attributed network embedding methods and attribute inference methods.
The four attributed network embedding baseline methods are:

- **GAE** [20]: GAE also embeds an attributed network by using the variational auto-encoder, but it optimizes the ELBO without considering the reconstruction error of attributes, and it learns embeddings for nodes only, rather than for both nodes and attributes.
- **GraphSAGE** [11]: This is a network embedding model that learns node representations by sampling and aggregating features from nodes' local neighborhoods.
- **AANE** [13]: It is an attributed network embedding model that learns node representations based on the decomposition of attribute affinities and the embedding difference between the connected nodes.
- **ANRL** [41]: It is a network embedding model that uses the neighbor enhancement auto-encoder and the attribute-aware skip-gram model to capture the node attributes and structural information of networks. We adopt one of its variants which uses the Weighted Average Neighbor function to construct its target neighbors, abbreviated as ANRL-WAN.

The three attribute inference baseline methods are:

- **SAN** [9]: This is a joint link prediction and attribute inference algorithm based on link features computed by Adamic-Adar and Low-Rank Approximation.
- **EdgeExp** [4]: This is an attribute inference algorithm that leverages a softmax function to solve for both user attributes and relationship types.
- **BLA** [40]: This is a probabilistic model that iteratively learns user links and attributes, and leverages the data redundancy on each side and the mutual reinforcement between the two.

We implemented all the baselines by the codes released by the authors. The parameters of all baselines are tuned to be optimal. For our model, the hyperparameters of prior distributions, $\bar{\sigma}^2_{\mathcal{V}}$,$\bar{\sigma}^2_{\mathcal{A}}$ (Eq. 6 and Eq. 7) are set to be 1 in all the experiments. We train our model for 200 iterations by Adam [17] optimizer with the learning rate being 0.01. For our inference neural network, we use a 64-dimensional hidden layer and 32-dimensional latent variables in all experiments. For fair comparisons, the dimension of embedding for all methods is fixed to be 32. Our CAN model [4] was implemented by TensorFlow [1].

Note that we did not take embedding methods for plain network as our baselines, as a large number of results show that embedding network by leveraging both the network structure and the associated node attribute can achieve better performance in majority of down-stream applications compared to approaches that only consider the topological structure [11, 13, 20].

6 RESULTS AND ANALYSIS

This section reports our experimental results. We first address **RQ1** by evaluating our CAN model on two graph mining tasks, i.e., link prediction and node classification. Then, to answer **RQ2**, we apply our CAN model on two more tasks, i.e., attribute inference and user profiling, which are unable to be addressed by existing attributed network embedding methods as they do not co-embed attributes and nodes. Finally, we visualize DBLP network layout by arranging both the node and attribute embeddings on 2-D space (**RQ3**).

6.1 Link Prediction

Link prediction, aiming at predicting if there exists an edge between two nodes, is a typical task in networks analysis. Following those in [20, 21], to evaluate the performance of our model, we randomly divide all edges into three sets, i.e., the training set (85%), the validating set (5%) and the testing set (10%). For negative instances (no edge between two nodes), we randomly sample an equal number of non-existing links. Both our CAN and GAE [20] reconstruct both positive and negative edges during training, and the predicted probability of a link can be achieved directly from the inner product of the embeddings between two nodes. For other baselines, we rank both positive and negative instances according to the cosine similarity between two nodes. For evaluation purpose, we employ area under the ROC curve (AUC) and average precision (AP) scores to evaluate the performance of link prediction, which is commonly used in related literature [20, 40]. Higher values of AUC and AP indicate better performance. Tab. 3 shows the link prediction performance of our CAN and the baselines on the six attributed networks. As shown, our CAN consistently performs better than any of the baseline model. In CORA, Citeseer, Facebook and Pubmed datasets, our CAN can even achieve higher than 95% AUC and AP scores, which indicates that our inferred embeddings work well in link prediction task. This is mainly because our CAN optimizes a loss function consisting of the reconstruction error of all the edges.

6.2 Node Classification

Node classification is another traditional task commonly used to evaluate the quality of the learned node embeddings. Similar to

[2] Available from: http://dblp.uni-trier.de/xml/.
[3] Available from: http://www.ccf.org.cn/.
[4] The code of our CAN model is publicly available from: https://github.com/mengzaiqiao/CAN.

Table 3: Link prediction performance. The best performing runs per metric per dataset are marked in boldface.

Method	Cora		Citeseer		Facebook		Pubmed		Flickr		BlogCatalog	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
AANE	.767	.720	.785	.765	.842	.834	.783	.754	.709	.697	.711	.714
GaphSAGE	.795	.763	.802	.791	.854	.846	.840	.829	.732	.728	.723	.702
ANRL-WAN	.832	.843	.867	.848	.935	.912	.918	.897	.724	.763	.762	.758
GAE	.914	.926	.908	.920	.980	.979	.944	.947	.828	.827	.821	.821
CAN	**.985**	**.984**	**.950**	**.958**	**.988**	**.986**	**.980**	**.977**	**.914**	**.922**	**.837**	**.837**

previous studies [13, 41], we employ Micro-F1 and Macro-F1 as metrics to measure the performance of node classification. After the node representations are obtained, we randomly sample 20% labeled nodes to train an SVM classifier and the rest are used for testing. We repeat this process for 10 times, and report the average performances on both Macro-F1 (Ma_F1) and Micro-F1 (Mi_F1). Tab. 4 shows the performance of our CAN and the baseline models in five labeled attributed networks. As shown in the table, our proposed CAN performs the best in Cora and Flickr datasets, and shows competitive performance in other datasets compared with other attributed network embedding baselines. The result of node classification task demonstrates that our CAN model can learn effective representations for nodes in attribute networks.

6.3 Attribute Inference

Attribute inference is a task that aims at predicting the value of attributes of the nodes. As using only the node embeddings cannot decode the affinities between nodes and attributes, the four baseline methods, i.e., AANE, GraphSAGE, ANRL-WAN and GAE, cannot be applied in this task. We take other three state-of-the-art attribute inference algorithms, i.e., SAN [9], EdgeExp [4] and BLA [40], as our baselines for performance comparison. We employ the AUC and AP metrics to measure the attribute inference performance. Tab. 5 presents the attribute inference performance of our CAN and the baseline models in the six attributed networks. We can find that our CAN model performs significantly better than all the baseline methods in all the datasets. The BLA method can always achieve the second best performance in this task. Moreover, the improvement between BLA and our CAN is always significant. This can be explained by the fact that our CAN optimizes a loss function consisting of the reconstruction error of all the attributes.

It is worth noting that our CAN model does not need any free parameters to trade the weights between topological structure and attributes. This highlights another important feature of CAN. It can achieve significantly improvement of performance in both link prediction and attribute inference tasks, **without task-specific fine-tuning**. Another fact validated by CAN is that attribute inference can help to improve link prediction; that is, link prediction accuracy is further improved by first inferring missing attributes [9]. In addition, all the methods show relatively poor performance on Pubmed dataset. However, our CAN model still scores the best. The facts making the attribute inference task in this dataset more challenging include: Pubmed has the largest node set but has a relatively small size of attribute set, and the ratio between the positive instances and the negative instances of nodes' attributes is only about 0.02, resulting in an unbalanced classification problem.

6.4 User Profiling

User profiling task, also known as expertise profiling task, aims at generating k keywords for profiling each user's expertise [25]. In academic social networks, authors' academic publications were often used to learn how personal research interests evolved [7] over time. In this setting, authors' publications indicate their expertise. In our experiment, we utilize the embeddings of nodes and attributes to obtain the most relevance attributes for each node as their expertise. Specifically, we first co-embed the DBLP academic social network into embeddings of authors and conferences. Then, we calculate the cosine similarities between authors and conferences according to their embeddings. Finally, we rank the conferences by the similarities to each author and return the top-4 conferences as the expertise for each author. Tab. 6 reports the profiling result for five renowned scholars. Among the five scholars, three of them (Michael I. Jordan, Geoffrey E. Hinton and Yann LeCun) are outstanding experts in the field of machine learning and the other two (Robert Endre Tarjan and Sanjeev Arora) are in the field of theoretical computer science. According to Tab. 6, the three machine learning experts are profiled by NIPS, ICML, UAI, etc., which are the top-tier machine learning conferences and the other two are profiled by SODA, PODC, STOC, etc., which are top-tier theoretical computer science conferences. We node that our model takes binary-valued attributes as input but we can still effectively retrieve top-4 relevant conferences based on the embeddings of authors and conferences. It also reveals that capturing affinities between nodes and attributes is essential for network embedding methods to characterize the node attributes.

6.5 Network Visualization

To qualitatively evaluate the result embeddings of our CAN by network visualization, we first obtain 2-dimensional Gaussian embeddings for authors and attributes of DBLP network. Then we plot their means and variances of the resulting embeddings into 2-dimensional space. For being clear in the figure, we show with embeddings of only 200 authors who own the top 200 high H-index [5] among others in the dataset. The visualization result is shown in Fig. 2, from which we have the following observations:

(1) The representations of conferences show larger variances than those of authors in general. This is because the inference model

[5]Please refer to http://www.guide2research.com/scientists/.

Table 4: Node classification performance. The best and the second best performing runs per metric per dataset are marked in boldface and underlined, respectively.

Method	Cora		Citeseer		Pubmed		Flickr		BlogCatalog	
	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1
AANE	.715	.720	.617	.671	.779	.814	.596	.615	.597	.617
GaphSAGE	.747	.763	.594	.630	.809	.817	.647	.652	.626	.643
ANRL-WAN	.774	.634	**.671**	.713	**.846**	**.847**	.673	.697	**.656**	**.679**
GAE	.773	.788	.582	.638	.823	.828	.591	.655	.622	.636
CAN	**.822**	**.838**	.642	**.720**	.829	.834	**.692**	**.701**	.646	.653

Table 5: Attribute inference performance. The best runs per metric per dataset are marked in boldface.

Method	Cora		Citeseer		Facebook		Pubmed		Flickr		BlogCatalog	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
EdgeExp	.682	.690	.707	.714	.671	.687	.586	.576	.678	.685	.684	.744
SAN	.664	.672	.679	.675	.712	.723	.579	.572	.653	.660	.694	.710
BLA	.808	.801	.854	.876	.868	.830	.622	.602	.730	.769	.787	.792
CAN	**.932**	**.916**	**.954**	**.939**	**.974**	**.971**	**.670**	**.652**	**.867**	**.865**	**.868**	**.867**

Table 6: Author profiling for example experts.

Experts	Top-4 conferences
Michael I. Jordan	NIPS, UAI, ICML, ICASSP
Geoffrey E. Hinton	NIPS, ICML, ICCV, ICASSP
Yann LeCun	NIPS, ICCV, ICML, CVPR
Robert Endre Tarjan	FOCS, STOC, COLT, SODA
Sanjeev Arora	SODA, FOCS, ICALP, COLT

for nodes takes both the adjacency matrix and the attribute matrix of nodes as input and uses the GCN [11, 21] to take the spectral convolutions of graphs into account during the training process, which is informative to describe a node in an attributed network. However, the inference model for attributes only takes the transposed adjacency matrix as input, which has relatively sparser feature than input feature of nodes, resulting in larger variance and uncertainty of their representations.

(2) Similar conferences have representations which are displayed closely in the same region on the 2-dimensional plane. Specifically, as seen in Fig. 2, SIGIR, AAAI, WSDM, WWW, NIPS, ICML AND SIGKDD, which are conferences of machine learning and its applications, are all plotted at the top region of the plane, while SODA, FOCS, STOC and ICALP, which are conferences of theoretical computer science, are all plotted at the bottom left region of the plane. This result indicates that our CAN model effectively captures the similarities between attributes.

(3) From Fig. 2, we can observe that author Michael I. Jordan is quite close to the seven machine learning conferences, and author Robert Endre Tarjan is quite close to the four theoretical computer science conferences. It is well-known that Michael I. Jordan is an expert on machine learning and Robert Endre Tarjan is an expert on theoretical computer science. Therefore this result validates that our CAN can be used to effectively measure the affinities between authors and conferences.

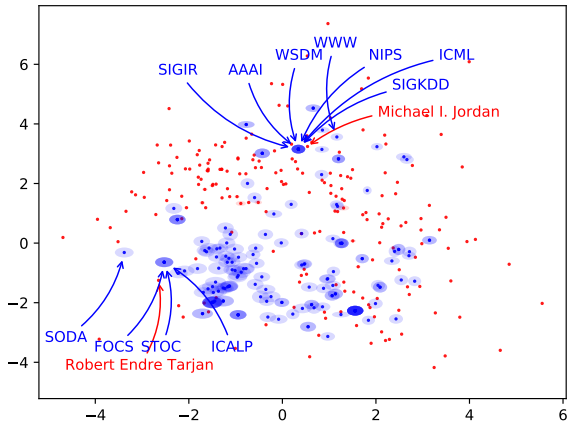

Figure 2: 2-dimensional visualization of Gaussian embeddings of authors and conferences for DBLP dataset. The red nodes are top 200 high H-index authors and the blue nodes are the 172 conferences in our DBLP dataset. The ellipsoids surrounding the nodes indicate variances of their embeddings. Some ellipsoids of conferences are in darker blue as these conferences are quite close and overlapped.

7 CONCLUSION

In this paper, to tackle the problem of embedding for attributed networks, we have proposed a novel co-embedding algorithm, called CAN. The proposed CAN learns the low-dimensional representations of both nodes and attributes in the same semantic space, such that the affinities between nodes and attributes of the networks can be effectively measured. To learn high-quality embeddings of both nodes and attributes in attributed networks, we have proposed a

variational auto-encoder algorithm, which consists of an inference model for encoding features of nodes and attributes into Gaussian distributions and a generative model for reconstructing both real and binary weighted edges and attributes. In our experiments, we have evaluated the performance of CAN and the baseline algorithms on seven real-world social networks. The experimental results have demonstrated that our CAN model is able to learn informative and high-quality representations of both nodes and attributes and significantly outperforms the state-of-the-art methods on several application tasks. Visualization on DBLP data shows that our CAN model properly maps nodes and attributes into a unified semantic space, where the affinities between authors and conferences can be preserved.

As to future work, we aim to extend our CAN model to a semi-supervised one, and dynamically embed nodes and attributes for attributed networks that evolve over time.

Acknowledgements

This work is supported by the King Abdullah University of Science and Technology (KAUST), Saudi Arabia.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, Vol. 16. 265–283.

[2] Aleksandar Bojchevski and Stephan GÃijnnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. *International Conference on Learning Representations*.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. ACM, 891–900.

[4] Deepayan Chakrabarti, Stanislav Funiak, Jonathan Chang, and Sofus A Macskassy. 2014. Joint inference of multiple label types in large networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32*. JMLR. org, II–874.

[5] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. 2016. Multilabel classification on heterogeneous graphs with gaussian embeddings. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 606–622.

[6] Alexey Dosovitskiy and Thomas Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*. 658–666.

[7] Yi Fang and Archana Godavarthy. 2014. Modeling the dynamics of personal expertise. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 1107–1110.

[8] Hongchang Gao and Heng Huang. 2018. Deep Attributed Network Embedding. In *International Joint Conference on Artificial Intelligence*. 3364–3370.

[9] Neil Zhenqiang Gong, Ameet Talwalkar, Lester Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Elaine (Runting) Shi, and Dawn Song. 2014. Joint Link Prediction and Attribute Inference Using a Social-Attribute Network. *ACM Trans. Intell. Syst. Technol* 5, 2 (Apr 2014), 27:1–27:20.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[12] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 623–632.

[13] Xiao Huang, Jundong Li, and Xia Hu. 2017. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 633–641.

[14] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 731–739.

[15] Xiao Huang, Qingquan Song, Jundong Li, and Xia Hu. 2018. Exploring expert cognition for attributed network embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 270–278.

[16] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 1965–1972.

[17] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[18] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*. 3581–3589.

[19] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.

[20] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning*.

[21] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*.

[22] Jure Leskovec and Julian J Mcauley. 2012. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*. 539–547.

[23] Jundong Li, Kewei Cheng, Liang Wu, and Huan Liu. 2018. Streaming link prediction on dynamic attributed networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 369–377.

[24] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 387–396.

[25] Shangsong Liang, Xiangliang Zhang, Zhaochun Ren, and Evangelos Kanoulas. 2018. Dynamic embeddings for user profiling in twitter. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1764–1773.

[26] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2016. Adversarial autoencoders. In *Advances in Neural Information Processing Systems*.

[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[28] Eric Nalisnick and Padhraic Smyth. 2017. Stick-breaking variational autoencoders. In *International Conference on Learning Representations*.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[30] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 459–467.

[31] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93.

[32] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*. 3483–3491.

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[34] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 817–826.

[35] Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. *International Conference on Learning Representations*.

[36] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[37] Suhang Wang, Charu Aggarwal, Jiliang Tang, and Huan Liu. 2017. Attributed signed network embedding. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 137–146.

[38] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *AAAI*. 203–209.

[39] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *International Joint Conference on Artificial Intelligence*. 2111–2117.

[40] Carl Yang, Lin Zhong, Li-Jia Li, and Luo Jie. 2017. Bi-directional Joint Inference for User Links and Attributes on Large Social Graphs. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 564–573.

[41] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. 2018. ANRL: Attributed Network Representation Learning via Deep Neural Networks. In *International Joint Conference on Artificial Intelligence*. 3155–3161.