Hyperspherical Variational Co-embedding for Attributed Networks

JINYUAN FANG, Sun Yat-sen University
SHANGSONG LIANG, Sun Yat-sen University, China and Mohamed bin Zayed University of Artificial Intelligence
ZAIQIAO MENG, University of Cambridge
MAARTEN DE RIJKE, University of Amsterdam

Network-based information has been widely explored and exploited in the information retrieval literature. Attributed networks, consisting of nodes, edges as well as attributes describing properties of nodes, are a basic type of network-based data, and are especially useful for many applications. Examples include user profiling in social networks and item recommendation in user-item purchase networks. Learning useful and expressive representations of entities in attributed networks can provide more effective building blocks to down-stream network-based tasks such as link prediction and attribute inference. Practically, input features of attributed networks are normalized as unit directional vectors. However, most network embedding techniques ignore the *spherical* nature of inputs and focus on learning representations in a Gaussian or Euclidean space, which, we hypothesize, might lead to less effective representations. To obtain more effective representations of attributed networks, we investigate the problem of mapping an attributed network with unit normalized directional features into a non-Gaussian and non-Euclidean space. Specifically, we propose a hyperspherical variational co-embedding for attributed networks (HCAN), which is based on generalized variational auto-encoders for heterogeneous data with multiple types of entities. HCAN jointly learns latent embeddings for both nodes and attributes in a unified hyperspherical space such that the affinities between nodes and attributes can be captured effectively. We argue that this is a crucial feature in many real-world applications of attributed networks. Previous Gaussian network embedding algorithms break the assumption of uninformative prior, which leads to unstable results and poor performance. In contrast, HCAN embeds nodes and attributes as von Mises-Fisher distributions, and allows one to capture the uncertainty of the inferred representations. Experimental results on eight datasets show that HCAN yields better performance in a number of applications compared with nine state-of-the-art baselines.

CCS Concepts: • **Human-centered computing** → *Social network analysis*; • **Theory of computation** → *Social networks;*

This work is supported by the National Natural Science Foundation of China (grant No. 61906219) and the Innovation Center for Artificial Intelligence (ICAI). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Authors' addresses: J. Fang, School of Computer Science and Engineering, Sun Yat-sen University, Outer Ring E Rd 132, Guangzhou 510006, China; email: fangjy6@gmail.com; S. Liang (corresponding author), School of Computer Science and Engineering, Sun Yat-sen University, Outer Ring E Rd 132, Guangzhou 510006, China, and Mohamed bin Zayed University of Artificial Intelligence, AI Diyafah St, Abu Dhabi 7909, United Arab Emirates; email: liangshangsong@gmail.com; Z. Meng, University of Cambridge, Trinity Ln, Cambridge CB2 1TN, The United Kingdom; email: zm324@cam.ac.uk; M. de Rijke, University of Amsterdam, Science Park 904, The Netherlands; email: m.derijke@uva.nl.

1046-8188/2021/12-ART58 $15.00
https://doi.org/10.1145/3478284

Additional Key Words and Phrases: Network embedding, generalized variational auto-encoders, hyperspherical representation

ACM Reference format:
Jinyuan Fang, Shangsong Liang, Zaiqiao Meng, and Maarten de Rijke. 2021. Hyperspherical Variational Co-embedding for Attributed Networks. *ACM Trans. Inf. Syst.* 40, 3, Article 58 (December 2021), 36 pages.
https://doi.org/10.1145/3478284

1 INTRODUCTION

Information is interconnected and therefore often represented as networks. Links between entities in networks represent the relationship between them and are highly informative for retrieval [50] and learning tasks [5]. For example, in networks of hyper-linked web documents, linked webpages are more likely to share the same topics even if their textual contents differ. Recently, a number of network embedding techniques have been proposed in order to effectively interpret and work with network data. They aim at inferring representations of entities in the network such as nodes, from which applications that use network data can benefit, including node classification [48, 56] and clustering [36, 37], community detection [18, 35], and link prediction [42]. These network embedding techniques map entities of a network into low-dimensional vectors in such a way that essential information about the whole network is preserved as much as possible, such as, e.g., topological structure and similarities between entities.

In this article, we focus on a basic type of network, namely *attributed networks*. Attributed networks do not only come with a topological structure, e.g., nodes and their connections, but also with rich attributes that describe properties of the nodes. For instance, in an academic search setting an attributed network could consist of authors (as entities), the co-authorship relation (as edges), and the journals/conferences the authors published at (as node attributes). Attributed networks are critical in many domains, ranging from online social networks to academic collaborative networks. Many embedding models have been proposed that learn a low-dimensional vector representations of nodes for attributed networks by leveraging both the topological structure and the attributes of the networks [20, 29, 33, 59].

Today's attributed network embedding algorithms suffer from a number of problems: (i) Most of them regard attributes as a single category of features of nodes, and focus on learning representations for nodes only but not for both attributes and nodes, as a result of which it is hard to measure similarities between nodes and attributes in the embedding space. Measuring such similarities is crucial to many attributed network applications, such as attribute inference for nodes [15] and user profiling [38]. (ii) Gaussians are the default choice as the prior for **variational auto-encoder** (**VAE**) based network embedding methods [12, 29]. This choice is mathematically convenient, but in many cases it breaks the assumption of an uninformative prior, which leads to unstable results and poor performance [9]. (iii) Most of the existing algorithms embed data into a Euclidean space. However, in data analysis, input features are usually normalized as unit vectors in a preprocessing step, i.e., distributing the input data on the unit sphere. For example, in citation networks where the features are bags-of-words, nodes or publications have typically been represented using a **term frequency-inverse document frequency** (**tf-idf**) normalized form, where each publication is represented as a point on a unit-sphere using a combination of both within document frequencies and inverse document frequencies. Such unit normalized feature vectors are directional features that remove the "magnitude" of the features and only keep the orientation of features as discriminative information. Hence, hyperplane representations in a Euclidean space may not be the appropriate representations concerning the directional features of the data. In fact, a variety of data, including texts, images, and sequential protein data are better represented through spherical representations rather than hyperplane representations [13, 41].

The VAE [28] offers a principled framework to learn representations for homogeneous data, but it has limited applicability in learning representations for heterogeneous data with multiple types of entities, such as the two types of entities, i.e., nodes and attributes, in attributed networks. To overcome this shortcoming, we propose a generalized variational auto-encoder for learning representations of heterogeneous data. Based on the generalized VAE framework, we propose the **hyperspherical variational co-embedding for attributed networks (HCAN)** to learn representations for attributed networks with unit normalized directional features. The HCAN co-embeds both nodes and attributes of the attributed networks in the same semantic space such that the affinities between them can be effectively measured. To effectively obtain embeddings of both nodes and attributes, hyperspherical variational co-embedding for attributed networks (HCAN) represents them as von Mises–Fisher distributions in hyperspherical space instead of Gaussian distributions. Utilizing the latent hyperspherical features of the data has been shown to be more effective than utilizing the hyperplane features of the data for many tasks, such as clustering for networks [1], face recognition [21], and text modeling [68]. Since the uniform distribution on the hypersphere is conveniently recovered as a special case of the von Mises–Fisher distribution, we can obtain a truly uninformative prior when placing a von Mises–Fisher prior on the embeddings of both nodes and attributes. Moreover, by representing the embeddings as von Mises–Fisher distributions, we are able to measure the uncertainty in the inferred representations with the concentration parameter assigned to each embedding. As the HCAN learns latent representations of both nodes and attributes in a unified hyperspherical space for the normalized directional data distributed on unit sphere, it can better capture latent embeddings that improve the performance of many attributed network tasks compared to models learning latent representations in Euclidean space.

To evaluate the effectiveness of the proposed hyperspherical co-embedding model, we conduct experiments on eight publicly available real-world datasets. For the purpose of evaluating the inferred node representations, we compare HCAN against baseline models in terms of the performance on node-oriented network tasks such as link prediction and node classification. For the purpose of evaluating the inferred attribute representations, we compare HCAN with baseline models in terms of the performance on attribute inference and user profiling tasks, where capturing similarities between nodes and attributes is the key to the success of the tasks. Our experimental results demonstrate that HCANs can achieve better performance and that inferring hyperspherical representations rather than hyperplane ones can improve performance.

The main contributions of our work can be summarized as follows:

(i) We generalize the variational auto-encoder (VAE) to learn hyperspherical representations for heterogeneous data with multiple types of entities and relations.
(ii) Based on the generalized VAE framework, we propose a hyperspherical co-embedding model, HCAN, to jointly learn low-dimensional embeddings of both nodes and attributes for attributed networks in a unified hyperspherical space such that the affinities between nodes and attributes can be captured and measured effectively.
(iii) The HCAN model maps entities of an attributed network, i.e., nodes and attributes, into a hyperspherical space by using the von Mises–Fisher distributions, which have been shown to be more effective in representing similarities between entities.
(iv) We design practical, trainable inference and generative networks to optimize the derived objective applied in the proposed HCAN model.
(v) We conduct extensive experiments on real-world attributed network datasets to evaluate the effectiveness of the learned embeddings, i.e., link prediction, node classification, attribute

inference and user profiling, and the results show that HCAN is able to learn high-quality network embeddings.

The remainder of this article is organized as follows. Section 2 discusses the related work. In Section 3, we define the hyperspherical co-embedding problem to be addressed. Section 4 provides a brief introduction to variational representation learning for homogeneous data in hyperplane space and then introduces generalized variational representation learning for heterogeneous data in hyperspherical space. Section 5 provides an overview of the variational co-embedding model for attributed networks and details the proposed model, HCAN. Section 6 describes our experimental setup and the experimental results are reported and analyzed in Section 7. Finally, Section 8 concludes the article.

2 RELATED WORK

In this section, we briefly discuss three lines of related work: network embeddings, VAE, and hyperspherical latent spaces. We start by surveying the use of attributed networks in information retrieval.

2.1 Attributed Networks in Information Retrieval

Real-world information can often be represented as attributed networks [50]. How to effectively explore and exploit this type of network-based information is a constant theme in the information retrieval literature. For example, user-item interactions in recommender systems can be considered as attributed networks where we treat users/items as nodes and the interactions between them as edges. A number of publications have proposed to learn representations of nodes in the user-item graphs [6, 61, 64, 65]. Wang et al. [64] propose **Neural Graph Collaborative Filtering** (**NGCF**), which explicitly incorporates a collaborative signal into the embedding model by leveraging high-order connectivities in the user-item integration graph. Chen et al. [6] propose the **Joint Neural Collaborative Filtering** (**J-NCF**) model that learns embeddings by leveraging both features of nodes and the interactions between nodes in the user-item graphs. Attributed network embedding techniques have also been widely applied in social network analysis [39, 62, 73]. For example, Zhao et al. [73] propose a network embedding approach to automatically generate tags for microblog users by leveraging rich social data. Online hyper-linked web documents can also be treated as attributed networks, where documents are regarded as nodes and the texts in documents are regarded as attributes. Yao et al. [70] propose a **Text Graph Convolutional Network** (**Text GCN**) to learn representations of documents by levering the linkage and attribute information, which can be applied in down-stream document classification tasks.

2.2 Network Embeddings

Learning representations of networks has gained much attention recently and many unsupervised learning methods have been proposed to embed networks as low-dimensional vectors. The learned representations produced by these methods are capable of preserving the structural and feature information of the original network while boosting the performance on down-stream tasks such as node classification [48, 56] and link prediction [42].

Some network embedding methods only rely on the topological structure of the network to learn useful embeddings, such as DeepWalk [48], node2vec [18], and **large-scale information network embedding** (**LINE**) [56], which learn representations based on random walks or edge sampling. These methods use skip-grams with a negative sampling neural network architecture originally proposed for word embeddings [45]. In addition, Wang et al. [63] propose to incorporate the community structure of the network into embedding methods.

In another line of work, some methods propose to incorporate auxiliary information, such as labels, node attributes, and text content, in the embedding model to obtain more useful embeddings of networks. Compared with embeddings based on network structure alone, the embeddings learned through these methods are able to capture semantic information of objects in the network. The **variational graph auto-encoder** (**VGAE**) introduced by Kipf and Welling [29] combines the VAE framework and a graph convolution network to obtain network embeddings by taking both the structure and the attribute information of the network as input. Hamilton et al. [20] propose the GraphSAGE model that learns node representations by sampling and aggregating features from the local neighborhood of nodes. Veličković et al. [59] introduce a masked self-attention mechanism into the network embedding model, which enables nodes to attend to their neighborhoods' features with different weights. Zhang et al. [72] and Gao and Huang [14] propose to learn node embeddings with customized deep neural network architectures, which aim at capturing the high degree non-linearity in both topological structure and attributes. The results show that combining different types of auxiliary information, instead of using structure information only, can provide different insights into embeddings of nodes. However, these models only learn embeddings of nodes instead of all entities in the network, i.e., nodes and attributes, and are not able to capture the uncertainty inherent in the embeddings.

Recently, some approaches have been proposed that embed nodes as distributions, which allow them to capture the uncertainty of the embeddings [2, 11, 22, 44]. The KG2E model introduced by He et al. [22] uses Gaussian distributions to represent each entity/relation of knowledge graphs. Dos Santos et al. [11] study heterogeneous graphs for node classification using Gaussian embeddings. As to embedding methods for attributed networks, Bojchevski and Günnemann [2] embed each node as Gaussian distributions according to the energy-based loss of a personalized ranking formulation.

In HCAN, we embed both nodes and attributes in a hyperspherical space using the von Mises–Fisher distributions, which allows us to capture the uncertainty with the concentration parameter of the von Mises–Fisher distribution.

2.3 Variational Auto-Encoders

VAEs [28, 52] are a type of powerful generative models that are able to effectively infer posterior distributions of latent variables from observational data based on the variational inference principle. For a generative model $p(\mathbf{x}|\mathbf{z})$ with observational data $\mathbf{x}$ and latent variable $\mathbf{z}$, it may be analytically intractable to infer the true posterior of the latent variable $p(\mathbf{z}|\mathbf{x})$ when the likelihood function is non-Gaussian. Variational inference approximates the intractable posterior with a simpler-form variational distribution such as a Gaussian distribution. To effectively reduce the number of variational parameters, the VAE proposes to replace the inference of latent variables with an inference neural network and the generative process of the observations with a generative neural network. The parameters of the inference network and the generative network are learned jointly by maximizing the evidence lower bound of the log marginal likelihood of observations. After optimization, the variational posterior over latent variables is inferred through the inference network.

Many variants of the standard VAE model have been introduced [17, 25, 27, 29, 55], which have been studied extensively and applied in generative models [3, 17, 55] and learning representations of data in unsupervised [25, 29] or semi-supervised [7, 27, 43] ways. Specifically, Gregor et al. [17] propose the **Deep Recurrent Attentive Writer** (**DRAW**) with sequential VAE framework that allows for the iterative construction of complex images. Liang et al. [38] propose a **dynamic user and word embedding model** (**DUWE**) that learns the representations of both users and words in the same semantic space such that the similarities between them can be measured effectively.

Meng et al. [43] propose a semi-supervised model that incorporates label information in the network embedding methods to learn more expressive representations.

Of special interest to us is the VGAE model, proposed by Kipf and Welling [29], which uses the VAE to learn representations for attributed networks as well. However, it only learns representations for nodes, rather than both nodes and attributes as we do in our model. Besides, it embeds the nodes in Euclidean space by inferring Gaussian distributions of latent variables, while we embed both the nodes and attributes in a hyperspherical space using the von Mises-Fisher distribution.

2.4 Hyperspherical Latent Spaces

Recently, models that exploit the latent hyperspherical manifold based on a von Misher–Fisher distribution have been widely studied. Gopal and Yang [16] propose a Bayesian von Mises–Fisher mixture model to improve the performance of clustering high-dimensional data and show that von Mises–Fisher-based clustering models can improve clustering performance over standard methods such as K-means and topic modeling empirically. Similarly, Hasnat et al. [21] utilize the vMF mixture model to learn representations of face pictures on the hyperspherical manifold, which can preserve the class information such that it can achieve satisfactory results on the face verification task. In addition, Reisinger et al. [51] propose to incorporate vMF distributions into **Latent Dirichlet Allocation** (**LDA**) models and introduce SAM, which models documents as directional distributions on an unit hypersphere while allowing a natural likelihood parameterization in terms of cosine distance.

There has also been work that combines embedding techniques and the vMF distribution to obtain powerful generative models [9, 19, 68]. This work shows that adding L2 normalization to the latent representations of deep autoencoder models during training, i.e., forcing the latent representations to distribute on a unit sphere, can greatly improve the performance of clustering [1]. Davidson et al. [9] propose the **hyperspherical variational auto-encoder** (**$\mathcal{S}$-VAE**), which has the same encoder-decoder structure as classic the VAE [28] but uses von Mises–Fisher distributions as both variational posteriors and prior distributions over latent variables as opposed to Gaussian distributions. However, it aims at inferring embeddings for only one category of entities in homogeneous data; it is still unknown how to obtain embeddings of two or more categories of entities in heterogeneous data, in our case, nodes and attributes in attributed networks. Xu and Durrett [68] use the hyperspherical variational auto-encoder for a text modeling task, where the model encodes the input text as latent von Mises–Fisher distributions and reconstructs the text using the decoder. But the model proposed in [68] treats the concentration parameter κ of the latent von Mises–Fisher distribution as a constant that remains the same for every approximate posterior distribution. As a result, the **Kullback-Leibler** (KL) divergence in the evidence lower bound is a constant and the model only optimizes the reconstruction term of the loss function, which limits the flexibility and the expressiveness of the latent distribution.

In our proposed co-embedding model, both the direction vector and the concentration parameter of the von Mises–Fisher distribution are treated as variables that are inferred with the encoder network.

3 PROBLEM DEFINITION

In this section, we first introduce the main notation used in the article, and then formalize the co-embedding problem for attributed networks.

3.1 Notation

We begin by introducing the main notation used in the article. We denote scalars with normal letters (e.g., N represents the number of nodes in the network), while sets are represented with

Table 1. Main Notation Used in the Article

Notation	Description
$\mathcal{G}$	an attributed network
$\mathcal{V}$	set of nodes
$\mathcal{A}$	set of attributes
$\mathcal{H}$	set of heterogeneous data
$\mathcal{E}$	set of entities in heterogeneous data
$\mathcal{X}$	set of features in heterogeneous data
$\mathcal{R}$	set of relations between entities in heterogeneous data
$\mathcal{S}$	set of observations in heterogeneous data
$N = \|\mathcal{V}\|$	number of nodes
$F = \|\mathcal{A}\|$	number of attributes
D	dimension of the latent variable
T	number of different types of entities in heterogeneous data
$\mathbf{A} \in \mathbb{R}^{N \times N}$	adjacency matrix of nodes
$\mathbf{X} \in \mathbb{R}^{N \times F}$	attribute matrix of nodes
$\mathbf{F}^{\mathcal{V}} \in \mathbb{R}^{N \times F}$	directional feature matrix of nodes
$\mathbf{F}^{\mathcal{A}} \in \mathbb{R}^{F \times N}$	directional feature matrix of attributes
$\mathbf{Z}^{\mathcal{V}} \in \mathbb{R}^{N \times D}$	latent representation matrix for all *nodes*
$\mathbf{Z}^{\mathcal{A}} \in \mathbb{R}^{F \times D}$	latent representation matrix for all *attributes*

calligraphy letters (e.g., $\mathcal{V}$ represents the set of nodes in the network). Vectors and matrices are denoted by lower case and bold letters (e.g., $\mathbf{z}$ represents the embedding vector of a node) and bold and uppercase letters (e.g., $\mathbf{A}$ represents the adjacency matrix), respectively. The ith row of a matrix is denoted with a subscript (e.g., $\mathbf{A}_i$), while a vector with a subscript represents a scalar element of that vector (e.g., $\mathbf{z}_i$ or $\mathbf{A}_{ij}$). The transpose of a matrix $\mathbf{A}$ is represented as $\mathbf{A}^\top$.

We define a network with a set of nodes and attributes to be an **attributed network**, denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathbf{A}, \mathbf{X})$. Here, $\mathcal{V}$ and $\mathcal{A}$ represent the set of nodes and attributes of the network, respectively, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of the network, and $\mathbf{X} \in \mathbb{R}^{N \times F}$ is the attribute matrix, where $N = |\mathcal{V}|$ and $F = |\mathcal{A}|$ are the numbers of nodes and attributes, respectively. We further use the attribute matrix and the transpose of the attribute matrix to represent the features of nodes and attributes, which is denoted as $\mathbf{F}^{\mathcal{V}} \triangleq \mathbf{X}$ and $\mathbf{F}^{\mathcal{A}} \triangleq \mathbf{X}^\top$, respectively. Note that we first normalize the feature vectors as unit length vectors, i.e., directional vectors distributed on a unit sphere, in a preprocessing step. We summarize our main notation in Table 1.

3.2 The Co-embedding Problem

The problem of learning representations for entities in attributed networks, i.e., both nodes and attributes, can be defined as follows. Given an attributed network $\mathcal{G}$, with sets of nodes $\mathcal{V}$ and attributes $\mathcal{A}$, adjacency matrix of nodes $\mathbf{A}$ and attribute matrix of nodes $\mathbf{X}$, we aim at learning latent representations of both nodes and attributes of the network, denoted as $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$, respectively. Here, $\mathbf{Z}^{\mathcal{V}} \in \mathbb{R}^{N \times D}$ and $\mathbf{Z}^{\mathcal{A}} \in \mathbb{R}^{F \times D}$ represent the latent representation matrices of all the nodes and attributes, respectively, while D represents the dimension of the latent representations and $D \ll F$. Concretely, the problem can be formulated as learning a function f that satisfies the following:

$$\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathbf{A}, \mathbf{X}) \xrightarrow{f} \mathbf{Z}^{\mathcal{V}}, \mathbf{Z}^{\mathcal{A}}.$$

The output of the function consists of the representations of both nodes and attributes of the network in the same semantic space. As a result, the learned representations $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$ allow us to preserve both the topological structure and the attribute information of attributed networks as much as possible while capturing the similarities between nodes and attributes.

4 VARIATIONAL REPRESENTATION LEARNING

VAEs [28, 52] are a type of generative models that offer a principled variational framework for learning representations from observations by jointly training a probabilistic encoder and a probabilistic decoder. The learned representations have been successfully applied in many critical domains such as item recommendation [66] and user profiling [38]. In this section, we first briefly review the variational representation learning for homogeneous data and then extend it to learn hyperspherical representations for heterogenous data. After that, we detail our motivation for learning representations in a hyperspherical space as opposed to a hyperplane space.

4.1 Variational Representation Learning for Homogeneous Data

We first review variational representation learning for homogeneous data that appears as $O = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^F$ is the feature vector of the ith data-point and F is the feature dimension. Following the principle of variational auto-encoders [28], the observation data are generated through a set of latent variables $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ with Gaussian priors. In the variational learning framework, the posteriors of these latent variables are considered as their representations. Since exact posterior inference of the latent variables is not necessarily analytically tractable, variational auto-encoders propose to approximate the true posteriors with simpler-form variational distributions such as Gaussian distributions. The variational distributions of latent variables are learned by maximizing the **evidence lower bound** (**ELBO**) of the log marginal likelihood of observations, which is given by

$$\mathcal{L}(O) = \sum_{i=1}^{n} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{x}_i)} [\log p_\theta(\mathbf{x}_i|\mathbf{z}_i)] - D_{KL}[q_\phi(\mathbf{z}_i|\mathbf{x}_i) \| p_\theta(\mathbf{z}_i)], \quad (1)$$

where $p_\theta(\mathbf{z}_i)$ is the prior of the latent variable $\mathbf{z}_i$, which is usually modeled by a standard multivariate Gaussian distribution, and $D_{KL}[\cdot\|\cdot]$ is the KL divergence between the variational posterior, which is also a Gaussian distribution, and the prior of the latent variable. The VAE framework consists of two networks to learn the variational distributions of latent variables: one is the inference network (also called the encoder) q_ϕ with parameter ϕ, which aims at inferring the variational posteriors over latent variables, while another one is the generative network (also called the decoder) p_θ with parameter θ, which reconstructs the observation based on the latent variables. The variational parameter ϕ and generative parameter θ are trained jointly using the stochastic gradient with the reparameterization trick [28]. After optimization, the representations of the homogeneous data-points can be obtained through the encoder network.

While the method is effective in learning representations for homogeneous data, which are generated independently according to a homogeneous prior, it cannot be directly applied in heterogeneous data, which consists of multiple types of entities and relations. Some approaches [34, 38] have been proposed to learn representations for multiple types of entities in heterogenous data based on the variational methods. However, the embeddings learned with these methods lie in the hyperplane space, which is represented as Gaussian distributions, as opposed to the hyperspherical space in this article. We leave a discussion of the limitations and solutions of Gaussian distributions as embeddings to Section 4.3. To the best of our knowledge, no model in the literature learns hyperspherical representations for heterogenous network data. Hence, in the next section, we propose a variational method that is able to learn hyperspherical representations for heterogenous data with arbitrary types of entities and relations.

4.2 Variational Representation Learning for Heterogeneous Data

We propose a generalized form of variational auto-encoders for heterogeneous data that appears as triplets. Let $\mathcal{H} = \{\mathcal{E}, \mathcal{X}, \mathcal{R}\}$ represent heterogeneous network data, where $\mathcal{E} = \{\mathcal{E}^1, \ldots, \mathcal{E}^T\}$ is the set of entities with multiple types (e.g., T types) associated with a set of features $\mathcal{X} = \{\mathbf{X}^1, \ldots, \mathbf{X}^T\}$. We use a superscript to denote features with a specific type, e.g., $\mathbf{X}^g$ represents the features of entities with type g. The set of relations between entities is denoted as $\mathcal{R} = \{r_{ij} | \mathbf{x}_i^g \in \mathbf{X}^g, \mathbf{x}_j^h \in \mathbf{X}^h\}$, where r_{ij} is the relationship between the two entities.

Without loss of generality, the observations of heterogeneous data $\mathcal{S}$ can be represented as a set of triplets $\mathcal{S}_{ij} = (\mathbf{x}_i^g, \mathbf{x}_j^h, r_{ij})$, where two entities may have the same or different types. We introduce a set of latent variables $\mathcal{Z} = \{\mathbf{Z}^1, \ldots, \mathbf{Z}^T\}$ corresponding to different types of entities. Let $\mathcal{Z}_{ij} = \{\mathbf{z}_i^g, \mathbf{z}_j^h\}$ be the collection of latent variables for the two entities in a triplet $\mathcal{S}_{ij}$ and $\mathcal{F}_{ij} = \{\mathbf{x}_i^g, \mathbf{x}_j^h\}$ be the feature vectors of the entities. By using Jensen's inequality, the ELBO of the log marginal likelihood of the triplet $\mathcal{S}_{ij}$ can be derived as:

$$\begin{aligned}\log p_\theta(\mathcal{S}_{ij}) &= \log \int p_\theta(\mathcal{S}_{ij}, \mathcal{Z}_{ij}) \mathrm{d}\mathcal{Z}_{ij} \\ &= \log \int p_\theta(\mathcal{S}_{ij}, \mathcal{Z}_{ij}) \frac{q_\phi(\mathcal{Z}_{ij}|\mathcal{F}_{ij})}{q_\phi(\mathcal{Z}_{ij}|\mathcal{F}_{ij})} \mathrm{d}\mathcal{Z}_{ij} \\ &\geqslant \mathbb{E}_{q_\phi(\mathcal{Z}_{ij}|\mathcal{F}_{ij})}\Big[\log p_\theta(\mathcal{S}_{ij}, \mathcal{Z}_{ij}) - \log q_\phi(\mathcal{Z}_{ij}|\mathcal{F}_{ij})\Big], \end{aligned} \tag{2}$$

where $p_\theta(\mathcal{S}_{ij}, \mathcal{Z}_{ij})$ is the joint distribution of the triplet, with θ being the generative parameters, and $q_\phi(\mathcal{Z}_{ij}|\mathcal{F}_{ij})$ is the variational distribution over entities approximating the true posterior of latent variables $p_\theta(\mathcal{Z}_{ij}|\mathcal{F}_{ij})$, with ϕ being the variational parameters to be estimated. Note that θ and ϕ may be different types depending on the heterogeneity of entities in the triplet. Following the standard VAE algorithm [28], we assume that the latent variables of the entities are independent and the joint probability distribution can be factorized as:

$$p_\theta(\mathcal{S}_{ij}, \mathcal{Z}_{ij}) = p_\theta\left(\mathbf{x}_i^g, \mathbf{x}_j^h, r_{ij}|\mathcal{Z}_{ij}\right) p\left(\mathbf{z}_i^g\right) p\left(\mathbf{z}_j^h\right). \tag{3}$$

In the setting of heterogeneous network data, we are more concerned about the reconstruction of relationships between entities than the features of entities. Hence, we can simplify the likelihood function of observation $p_\theta(\mathbf{x}_i^g, \mathbf{x}_j^h, r_{ij}|\mathcal{Z}_{ij})$ as $p_\theta(r_{ij}|\mathcal{Z}_{ij})$. As a result, by substituting Equation (3) in Equation (2), Equation (2) can can denoted as:

$$\begin{aligned}\log p_\theta(\mathcal{S}_{ij}) &\geqslant \mathbb{E}_{q_\phi(\mathcal{Z}_{ij}|\mathcal{F}_{ij})}\Big[\log p_\theta(r_{ij}|\mathcal{Z}_{ij})\Big] - D_{KL}\Big[q_\phi(\mathcal{Z}_{ij}|\mathcal{F}_{ij}) \| p\left(\mathbf{z}_i^g\right) p\left(\mathbf{z}_j^h\right)\Big] \\ &\triangleq \mathcal{L}(\mathcal{S}_{ij}), \end{aligned} \tag{4}$$

where we use $\mathcal{L}(\mathcal{S}_{ij})$ to represent the ELBO of the triplet $\mathcal{S}_{ij}$. For the overall observations of heterogeneous data $\mathcal{S}$, the evidence lower bound can be written as:

$$\mathcal{J}(\mathcal{S}) = \sum_{\mathcal{S}_{ij} \in \mathcal{S}} \mathcal{L}(\mathcal{S}_{ij}). \tag{5}$$

From the perspective of auto-encoders, q_ϕ is the probabilistic encoder that encodes the features of entities $\mathcal{F}_{ij}$ as variational distribution over representations of entities, sampling from which we can obtain the latent representations, while the p_θ is the probabilistic decoder that takes the representations of entities as input and reconstructs the relationship between the entities. The KL-divergence can be seen as a regularization term that encourages the variational distributions to be close to the priors. The generative parameters θ and variational parameters ϕ can be optimized by maximizing the overall evidence lower bound (i.e., Equation (5)).

In the classic VAE [28, 29] setting, we obtain representations in a hyperplane space by choosing both the prior and the variational posterior to be a Gaussian distribution. Davidson et al. [9] propose a hyperspherical VAE ($\mathcal{S}$-VAE) that is able to learn representations in hyperspherical space for homogeneous data. This is achieved by replacing the Gaussian distribution in the VAE framework with a von Mises-Fisher distribution, which is a probability distribution defined on the sphere. We can easily integrate the $\mathcal{S}$-VAE in our generalized VAE for heterogeneous data by setting the prior and the variational posterior of latent variables to be a von Mises–Fisher distribution and hence we can obtain hyperspherical representations for the entities in heterogeneous network data.

4.3 Motivation for Embeddings in Hyperspherical Space

As mentioned above, we aim at learning representations of entities in heterogeneous data in a hyperspherical space instead of a hyperplane space. We achieve this by inferring the entities' latent von Mises–Fisher distributions instead of their Gaussian distributions given the observations. Although a Gaussian distribution is mathematically convenient, it exhibits problematic properties: (i) If we embed the original data into a lower dimension, the Gaussian density presents a concentrated probability mass around the origin due to the KL-divergence of the ELBO, causing the data to cluster around the origin. This is particularly problematic when we want to divide the data into multiple clusters. An ideal prior/posterior should only stimulate the variance of the prior and posterior without forcing its mean to be close to the center. Priors/posteriors satisfying these properties are a uniform distribution over the entire space [9, 16, 19, 21]. However, such a uniform prior is not well defined on the hyperplane. (ii) If we embed the original data into higher dimensions, a Gaussian distribution of the embedded data in high dimensions tends to resemble a uniform distribution on the surface of a hypersphere, with most of its mass concentrated on the hyperspherical shell. Thus, it is natural to define the prior/posterior to distribute on a hypersphere rather than a hyperplane space. (iii) This is also motivated from a theoretical point of view, since the Gaussian definition is based on the L-2 norm that suffers from the curse of dimensionality. (iv) In practice, it has been shown that utilizing embeddings inferred in hyperspherical space instead of embeddings in hyperplane space (especially those that are modeled as Gaussian distributions) helps to boost the performance of many applications, e.g., image classification [8] and video completion [67].

5 HYPERSPHERICAL VARIATIONAL CO-EMBEDDING

In this section, we apply the generalized VAE for heterogeneous data (Section 4.2) to learn hyperspherical representations for entities in attributed networks with unit length directional feature vectors. Attributed networks can be considered as a kind of heterogeneous data where the nodes and attributes are two types of entities, i.e., $\mathcal{E} = \{\mathcal{V}, \mathcal{A}\}$, and the adjacency matrix and attribute matrix represent the relationships between the entities, i.e., $\mathcal{R} = \{\mathbf{A}, \mathbf{X}\}$. The set of features in an attributed network is given by $\mathcal{X} = \{\mathbf{F}^{\mathcal{V}}, \mathbf{F}^{\mathcal{A}}\}$, where $\mathbf{F}^{\mathcal{V}}$ and $\mathbf{F}^{\mathcal{A}}$ are the unit normalized directional features of nodes and attributes, respectively. Furthermore, the set of latent variables is denoted as $\mathcal{Z} = \{\mathbf{Z}^{\mathcal{V}}, \mathbf{Z}^{\mathcal{A}}\}$, where $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$ represent the embeddings for nodes and attributes, respectively.

To address the co-embedding problem, we propose a hyperspherical variational co-embedding model (abbreviated as HCAN), that co-embeds nodes and attributes of attributed networks. The goal of HCAN is to embed nodes and attributes of the attributed network in the same hyperspherical space and this can be achieved by mapping the entities as von Mises–Fisher distributions, which serve as the latent embeddings of both nodes and attributes. The learned embeddings are able to preserve the structure and attribute information of the original network as much

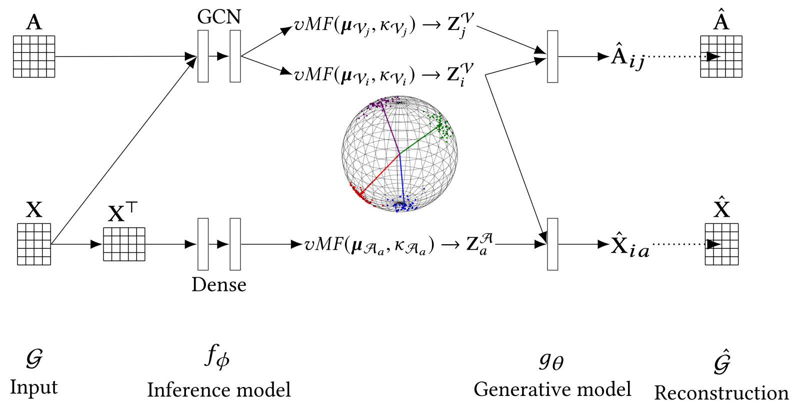

A
GCN
$vMF(\mu_{\mathcal{V}_j}, \kappa_{\mathcal{V}_j}) \rightarrow Z_j^{\mathcal{V}}$
$vMF(\mu_{\mathcal{V}_i}, \kappa_{\mathcal{V}_i}) \rightarrow Z_i^{\mathcal{V}}$
$\hat{A}_{ij}$
$\hat{A}$
X
$X^\top$
Dense
$vMF(\mu_{\mathcal{A}_a}, \kappa_{\mathcal{A}_a}) \rightarrow Z_a^{\mathcal{A}}$
$\hat{X}_{ia}$
$\hat{X}$
$\mathcal{G}$
Input
f_ϕ
Inference model
g_θ
Generative model
$\hat{\mathcal{G}}$
Reconstruction

Fig. 1. Architecture of the proposed HCAN. The inference network f_ϕ and the generative network g_θ are the probabilistic encoder and decoder of the proposed model, respectively. The encoder of the model takes the adjacency matrix **A** and attribute matrix **X** as input and embeds them into a unified hyperspherical space by outputing the direction vectors and the concentraction parameters of the von Mises–Fisher distributions over latent representations, while the decoder of the model reconstructs the input based on the latent representations.

as possible. Moreover, by representing the embeddings as von Mises–Fisher distributions, the variances of the distributions enable us to capture the uncertainty inherent in the network, which represents the noise in the network data caused by errors in data collecting and processing steps.

In what follows, we first provide an overview of the proposed model and then we briefly introduce the von Mises–Fisher distribution. Then we follow the principle of generalized VAEs for heterogeneous data (Section 4.2) to derive the overall evidence lower bound for the co-embedding problem of attributed networks. After that, since the variational distributions over latent variables are von Mises–Fisher distributions, we describe KL divergence, sampling and reparameterization problems to enable efficient optimization. Finally, we provide a detailed description of the inference and generative networks of our model.

5.1 Overview

Standard VAE algorithms [28] contain an inference network that maps an entity to a corresponding latent random variable and a generative network that reconstructs the input based on the latent variables. Inspired by standard VAEs, the proposed hyperspherical co-embedding model for attributed networks contains three main components, as illustrated in Figure 1: (i) An inference network for nodes that takes both the adjacency matrix **A** and feature matrix of nodes $\mathbf{F}^{\mathcal{V}}$ as input and outputs the approximate posterior distributions over representations of nodes. (ii) An inference network for attributes that outputs the approximate posterior distributions over representations of attributes based on the feature matrix of attributes $\mathbf{F}^{\mathcal{A}}$. In contrast to the traditional VAE and its variants, as illustrated in Section 4.3, the approximate posterior distributions over both the representations of nodes and attributes are von Mises–Fisher distributions instead of factorized Gaussian distributions, which allows us to obtain latent representations in hyperspherical space while better preserving the structure and attribute information of the network. (iii) A generative network that takes both the latent embeddings of the nodes and attributes as input and tries to reconstruct the adjacency matrix and attribute matrix.

Following the principle of VAEs, the parameters of both the inference networks and the generative network are trained jointly by maximizing the ELBO of the log marginal likelihood of

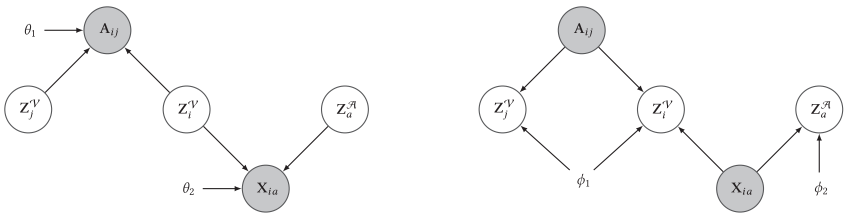


Fig. 2. Graphical representations of HCAN. The left subfigure describes the generative process, while the right subfigure describes the inference process.

the observations. Moreover, the gradients with respect to the parameters are computed with the reparameterization trick proposed by Naesseth et al. [46].

5.2 Von Mises–Fisher Distribution

The von Mises–Fisher distribution is a probability distribution defined on the $(D-1)$-dimensional sphere in $\mathbb{R}^D$. It is parameterized by $\boldsymbol{\mu} \in \mathbb{R}^D$ and $\kappa \in \mathbb{R}^+$, representing the mean direction vector and the concentration parameter, respectively. The probability density function of a von Mises–Fisher distribution for a D dimensional random unit vector $\mathbf{z} \in \mathbb{R}^D$ is defined as:

$$q(\mathbf{z}|\boldsymbol{\mu}, \kappa) = C_D(\kappa) \exp(\kappa \boldsymbol{\mu}^\top \mathbf{z}), \tag{6}$$

$$C_D(\kappa) = \frac{\kappa^{D/2-1}}{(2\pi)^{D/2} \mathcal{I}_{D/2-1}(\kappa)}, \tag{7}$$

where $\|\boldsymbol{\mu}\| = 1$ and $C_D(\kappa)$ is the normalizing constant where $\mathcal{I}_v$ stands for the modified Bessel function of the first kind at order v [10].

A high concentration parameter κ of the von Mises–Fisher distribution often leads to a tighter distribution around $\boldsymbol{\mu}$, the mean and the mode of the distribution. For the case when $\kappa = 0$, the von Mises–Fisher distribution becomes a uniform distribution over the sphere independent of the mean direction $\boldsymbol{\mu}$.

We utilize the von Mises–Fisher distribution as both the prior and variational distribution of the latent variables in our model. Specifically, HCAN uses a von Mises–Fisher distribution with concentration parameter $\kappa = 0$, which is a uniform distribution on a hypersphere, as prior distribution over the latent variable $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$. Since the true posterior over latent variables $p(\mathbf{Z}^{\mathcal{V}}, \mathbf{Z}^{\mathcal{A}}|\mathbf{X}, \mathbf{A})$ is intractable, we approximate it with variational distributions $q(\mathbf{Z}^{\mathcal{V}}|\mathbf{F}^{\mathcal{V}})$ and $q(\mathbf{Z}^{\mathcal{A}}|\mathbf{F}^{\mathcal{A}})$, which are both von Mises–Fisher distributions.

5.3 Variational Evidence Lower Bound

To embed nodes and attributes of the attribute network $\mathcal{G}$ in hyperspherical space, we first derive an objective function to be optimized, which is the overall evidence lower bound of the observations $\mathcal{S}$, i.e., the adjacency matrix $\mathbf{A}$ and the attribute matrix $\mathbf{X}$. Following the principle of generalized VAEs for heterogeneous data, we derive the lower bound by splitting the observations into two types of triplets. Specifically, the elements in observation matrices can be categorized into two types of triplet: (i) the edges connecting two nodes, which can be represented as $\mathcal{S}_{ij}^{\mathcal{V}} = (\mathbf{F}_i^{\mathcal{V}}, \mathbf{F}_j^{\mathcal{V}}, \mathbf{A}_{ij})$, where $\mathbf{F}_i^{\mathcal{V}}$ is the feature vector of node $\mathcal{V}_i$ and similar for $\mathbf{F}_j^{\mathcal{V}}$; (ii) the attribute values between nodes and attributes, which can be represented as $\mathcal{S}_{ia}^{\mathcal{A}} = (\mathbf{F}_i^{\mathcal{V}}, \mathbf{F}_a^{\mathcal{A}}, \mathbf{X}_{ia})$, where $\mathbf{F}_a^{\mathcal{A}}$ is the feature vector of the attribute $\mathcal{A}_a$. The overall generative and inference processes are given by the probabilistic graph shown in Figure 2.

Case 1: Edges connecting two nodes. For an edge $\mathbf{A}_{ij}$ between node $\mathcal{V}_i$ and node $\mathcal{V}_j$, we denote the collection of latent variables associated with the two nodes as $\mathcal{Z}_{ij}^{\mathcal{V}} = \{\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}\}$ and the collection of feature vectors as $\mathcal{F}_{ij}^{\mathcal{V}} = \{\mathbf{F}_i^{\mathcal{V}}, \mathbf{F}_j^{\mathcal{V}}\}$. We introduce a variational distribution over latent variables $\mathcal{Z}_{ij}^{\mathcal{V}}$ conditioned on $\mathcal{F}_{ij}^{\mathcal{V}}$ and by using Jensen's inequality, we obtain the lower bound for the logarithm marginal likelihood of the triplet:

$$\log p_\theta\left(\mathcal{S}_{ij}^{\mathcal{V}}\right) \geqslant \mathbb{E}_{q_\phi\left(\mathcal{Z}_{ij}^{\mathcal{V}}|\mathcal{F}_{ij}^{\mathcal{V}}\right)}\left[\log p_\theta\left(\mathcal{S}_{ij}^{\mathcal{V}}, \mathcal{Z}_{ij}^{\mathcal{V}}\right) - \log q_\phi\left(\mathcal{Z}_{ij}^{\mathcal{V}}|\mathcal{F}_{ij}^{\mathcal{V}}\right)\right], \tag{8}$$

where $q_\phi(\mathcal{Z}_{ij}^{\mathcal{V}}|\mathcal{F}_{ij}^{\mathcal{V}})$ is the variational distribution that approximates the true posterior of latent variables, with ϕ representing the parameters of the inference process, and $p_\theta(\mathcal{S}_{ij}^{\mathcal{V}}, \mathcal{Z}_{ij}^{\mathcal{V}})$ is the joint distribution of edges, with θ representing the parameters of the generative process. The joint distribution can be further written as:

$$p_\theta\left(\mathcal{S}_{ij}^{\mathcal{V}}, \mathcal{Z}_{ij}^{\mathcal{V}}\right) = p_{\theta_1}\left(\mathbf{A}_{ij}|\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}\right) p\left(\mathbf{Z}_i^{\mathcal{V}}\right) p\left(\mathbf{Z}_j^{\mathcal{V}}\right), \tag{9}$$

where θ_1 are the parameters of the generative model for edges and $p(\mathbf{Z}_i^{\mathcal{V}})$ is the prior over embeddings of node $\mathcal{V}_i$, similar for $p(\mathbf{Z}_j^{\mathcal{V}})$. As for the variational distribution $q_\phi(\mathcal{Z}_{ij}^{\mathcal{V}}|\mathcal{F}_{ij}^{\mathcal{V}})$, by using the mean-field assumption, we can factorize it as:

$$q_\phi\left(\mathcal{Z}_{ij}^{\mathcal{V}}|\mathcal{F}_{ij}^{\mathcal{V}}\right) = q_{\phi_1}\left(\mathbf{Z}_i^{\mathcal{V}}|\mathbf{F}_i^{\mathcal{V}}\right) q_{\phi_1}\left(\mathbf{Z}_j^{\mathcal{V}}|\mathbf{F}_j^{\mathcal{V}}\right), \tag{10}$$

where ϕ_1 are the parameters of the inference model for nodes. As a result, by combining the joint distribution (Equation (9)) and the variational distribution (Equation (10)) in the lower bound (Equation (8)), the lower bound is represented as:

$$\begin{aligned}\log p_\theta\left(\mathcal{S}_{ij}^{\mathcal{V}}\right) \geqslant\ & \mathbb{E}_{q_\phi\left(\mathcal{Z}_{ij}^{\mathcal{V}}|\mathcal{F}_{ij}^{\mathcal{V}}\right)}\left[\log p_{\theta_1}\left(\mathbf{A}_{ij}|\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}\right)\right] - D_{KL}\left[q_{\phi_1}\left(\mathbf{Z}_i^{\mathcal{V}}|\mathbf{F}_i^{\mathcal{V}}\right) \| p\left(\mathbf{Z}_i^{\mathcal{V}}\right)\right] \\ & - D_{KL}\left[q_{\phi_1}\left(\mathbf{Z}_j^{\mathcal{V}}|\mathbf{F}_j^{\mathcal{V}}\right) \| p\left(\mathbf{Z}_j^{\mathcal{V}}\right)\right] \\ & \triangleq \mathcal{U}\left(\mathcal{S}_{ij}^{\mathcal{V}}\right),\end{aligned} \tag{11}$$

where we use $\mathcal{U}(\mathcal{S}_{ij}^{\mathcal{V}})$ to denote the ELBO of the marginal likelihood of observation $\mathcal{S}_{ij}^{\mathcal{V}}$, and $D_{KL}[\cdot\|\cdot]$ is the KL divergence.

Case 2: Attribute values between nodes and attributes. In this case, $\mathbf{X}_{ia}$ represents the attribute value between node $\mathcal{V}_i$ and attribute $\mathcal{A}_a$. Let $\mathcal{Z}_{ia}^{\mathcal{A}} = \{\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}\}$ be the collection of latent variables and $\mathcal{F}_{ia}^{\mathcal{A}} = \{\mathbf{F}_i^{\mathcal{V}}, \mathbf{F}_a^{\mathcal{A}}\}$ be the collection of feature vectors associated with the triplet $\mathcal{S}_{ia}^{\mathcal{A}} = (\mathbf{F}_i^{\mathcal{V}}, \mathbf{F}_a^{\mathcal{A}}, \mathbf{X}_{ia})$. Then the lower bound of the logarithm marginal likelihood of the triplet can be written as:

$$\log p_\theta\left(\mathcal{S}_{ia}^{\mathcal{A}}\right) \geqslant \mathbb{E}_{q_\phi\left(\mathcal{Z}_{ia}^{\mathcal{A}}|\mathcal{F}_{ia}^{\mathcal{A}}\right)}\left[\log p_\theta\left(\mathcal{S}_{ia}^{\mathcal{A}}, \mathcal{Z}_{ia}^{\mathcal{A}}\right) - \log q_\phi\left(\mathcal{Z}_{ia}^{\mathcal{A}}|\mathcal{F}_{ia}^{\mathcal{A}}\right)\right], \tag{12}$$

where the joint probability distribution $p_\theta(\mathcal{S}_{ia}^{\mathcal{A}}, \mathcal{Z}_{ia}^{\mathcal{A}})$ can be represented as:

$$p_\theta\left(\mathcal{S}_{ia}^{\mathcal{A}}, \mathcal{Z}_{ia}^{\mathcal{A}}\right) = p_{\theta_2}\left(\mathbf{X}_{ia}|\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}\right) p\left(\mathbf{Z}_i^{\mathcal{V}}\right) p\left(\mathbf{Z}_a^{\mathcal{A}}\right), \tag{13}$$

where θ_2 are the parameters of the generative model for attributes. We can further factorize the variational distribution $q_\phi(\mathcal{Z}_{ia}^{\mathcal{A}}|\mathcal{F}_{ia}^{\mathcal{A}})$ as:

$$q_\phi\left(\mathcal{Z}_{ia}^{\mathcal{A}}|\mathcal{F}_{ia}^{\mathcal{A}}\right) = q_{\phi_1}\left(\mathbf{Z}_i^{\mathcal{V}}|\mathbf{F}_i^{\mathcal{V}}\right) q_{\phi_2}\left(\mathbf{Z}_a^{\mathcal{A}}|\mathbf{F}_a^{\mathcal{A}}\right), \tag{14}$$

where ϕ_2 are the parameters of the inference model for attributes. Substituting Equations (13) and (14) in Equation (12), Equation (12) can be written as:

$$\begin{aligned}\log p_\theta\left(\mathcal{S}_{ia}^{\mathcal{A}}\right) \geqslant\ & \mathbb{E}_{q_\phi(\mathcal{Z}_{ia}^{\mathcal{A}}|\mathcal{F}_{ia}^{\mathcal{A}})}\left[\log p_{\theta_2}\left(\mathbf{X}_{ia}|\mathbf{Z}_i^{\mathcal{V}},\mathbf{Z}_a^{\mathcal{A}}\right)\right] - D_{KL}\left[q_{\phi_1}\left(\mathbf{Z}_i^{\mathcal{V}}|\mathbf{F}_i^{\mathcal{V}}\right)\|p\left(\mathbf{Z}_i^{\mathcal{V}}\right)\right]\\ & - D_{KL}\left[q_{\phi_2}\left(\mathbf{Z}_a^{\mathcal{A}}|\mathbf{F}_a^{\mathcal{A}}\right)\|p\left(\mathbf{Z}_a^{\mathcal{A}}\right)\right]\\ \triangleq\ & \mathcal{M}\left(\mathcal{S}_{ia}^{\mathcal{A}}\right),\end{aligned} \tag{15}$$

where we use $\mathcal{M}(\mathcal{S}_{ia}^{\mathcal{A}})$ to denote the ELBO of marginal likelihood of observation $\mathcal{S}_{ia}^{\mathcal{A}}$.
Since the ELBOs for the two types of observations are available, we can obtain the overall evidence lower bound on the marginal likelihood for the entire adjacency matrix and attribute matrix, which is given by

$$\mathcal{J}(\mathbf{A},\mathbf{X}) = \sum_{\mathbf{A}_{ij}\in\mathbf{A}}\mathcal{U}\left(\mathcal{S}_{ij}^{\mathcal{V}}\right) + \sum_{\mathbf{X}_{ia}\in\mathbf{X}}\mathcal{M}\left(\mathcal{S}_{ia}^{\mathcal{A}}\right). \tag{16}$$

In the objective function of Equation (16), the left term is the loss of the adjacency matrix, while the right term is the loss of the attribute matrix. As a result, the learned latent representations of nodes and attributes are capable of capturing both the network structure and node attribute information. However, to make our model more flexible in controlling the loss between edges and attributes, we introduce a free parameter α to govern the trade-off between reconstruction accuracy of edges and attributes. Therefore, the final objective function of the proposed model, HCAN, is as follows:

$$\mathcal{J}(\mathbf{A},\mathbf{X}) = \alpha\cdot\left(\sum_{\mathbf{A}_{ij}\in\mathbf{A}}\mathcal{U}\left(\mathcal{S}_{ij}^{\mathcal{V}}\right)\right) + (1-\alpha)\cdot\left(\sum_{\mathbf{X}_{ia}\in\mathbf{X}}\mathcal{M}\left(\mathcal{S}_{ia}^{\mathcal{A}}\right)\right). \tag{17}$$

The overall generative parameters $\theta = \{\theta_1,\theta_2\}$ and variational parameters $\phi = \{\phi_1,\phi_2\}$ in HCAN are learned by maximizing the objective function of Equation (17). As mentioned before, we argue that the structure and attribute information of the attributed network can be better preserved in hyperspherical space than in hyperplane space, so we choose the prior and the variational distributions of latent variables to be von Mises–Fisher distributions which can be represented by

$$p\left(\mathbf{Z}_i^{\mathcal{V}}\right) = U(S^{D-1}), \tag{18}$$

$$p\left(\mathbf{Z}_a^{\mathcal{A}}\right) = U(S^{D-1}), \tag{19}$$

$$q_{\phi_1}\left(\mathbf{Z}_i^{\mathcal{V}}|\mathbf{F}_i^{\mathcal{V}}\right) = \mathrm{vMF}\left(\mathbf{Z}_i^{\mathcal{V}}|\boldsymbol{\mu}_{\phi_1}\left(\mathbf{F}_i^{\mathcal{V}}\right),\kappa_{\phi_1}\left(\mathbf{F}_i^{\mathcal{V}}\right)\right), \tag{20}$$

$$q_{\phi_2}\left(\mathbf{Z}_a^{\mathcal{A}}|\mathbf{F}_a^{\mathcal{A}}\right) = \mathrm{vMF}\left(\mathbf{Z}_a^{\mathcal{A}}|\boldsymbol{\mu}_{\phi_2}\left(\mathbf{F}_a^{\mathcal{A}}\right),\kappa_{\phi_2}\left(\mathbf{F}_a^{\mathcal{A}}\right)\right), \tag{21}$$

where $U(S^{D-1})$ represents the von Mises–Fisher distribution with concentration parameter $\kappa = 0$, which is a uniform distribution defined on the $(D-1)$ dimensional sphere and D is the dimension of the latent variables. The parameters of the variational distribution over latent variables of nodes $q_{\phi_1}(\mathbf{Z}_i^{\mathcal{V}}|\mathbf{F}_i^{\mathcal{V}})$ are computed with non-linear functions $\boldsymbol{\mu}_{\phi_1}$ and κ_{ϕ_1} by taking the feature vector of nodes as input. Similarly, $\boldsymbol{\mu}_{\phi_2}$ and κ_{ϕ_2} are non-linear functions used to compute the parameters of the variational distribution of attributes $q_{\phi_2}(\mathbf{Z}_a^{\mathcal{A}}|\mathbf{F}_a^{\mathcal{A}})$.

5.4 KL Divergence

In order to maximize the objective function of Equation (17), we need to derive an expression to compute the KL divergence between the approximate posterior and the prior, both of which are von Mises–Fisher distributions. The KL divergence between a von Mises–Fisher distribution

$q(\mathbf{z}|\boldsymbol{\mu}_\phi, \kappa_\phi)$ and a uniform distribution defined on the hypersphere

$$p(\mathbf{z}) = \left(\frac{2(\pi^{D/2})}{\Gamma(D/2)}\right)^{-1}, \tag{22}$$

which is given by one divided by the surface area of $\mathcal{S}^{(D-1)}$, can be represented as:

$$\mathrm{D_{KL}}[q(\mathbf{z}|\boldsymbol{\mu}_\phi, \kappa_\phi)\|p(\mathbf{z})] = \kappa_\phi \frac{\mathcal{I}_{D/2}(\kappa_\phi)}{\mathcal{I}_{D/2-1}(\kappa_\phi)} + \left(\frac{D}{2} - 1\right)\log \kappa_\phi - \left(\frac{D}{2}\right)\log(2\pi)$$
$$- \log \mathcal{I}_{D/2-1}(\kappa_\phi) + \left(\frac{D}{2}\right)\log \pi + \log 2 - \log \Gamma\left(\frac{D}{2}\right),$$

where $\Gamma(x)$ is the gamma function. Note that the KL divergence depends only on the concentration parameter κ_ϕ of the von Mises–Fisher distribution and not on the direction vector $\boldsymbol{\mu}_\phi$. Besides, as we treat the parameter κ_ϕ as a variable computed by the encoder, we need to compute the gradient of the KL divergence with respective to the variational parameter ϕ, which can be represented as:

$$\begin{aligned} &\nabla_\phi \mathrm{D_{KL}}\left[q(\mathbf{z}|\boldsymbol{\mu}_\phi, \kappa_\phi)\|p(\mathbf{z})\right] \\ &= \nabla_\phi \kappa_\phi \nabla_{\kappa_\phi} \mathrm{D_{KL}}\left[q(\mathbf{z}|\boldsymbol{\mu}_\phi, \kappa_\phi)\|p(\mathbf{z})\right] \\ &= \nabla_\phi \kappa_\phi \cdot \left(\frac{1}{2}\kappa_\phi\left(\frac{\mathcal{I}_{D/2+1}(\kappa_\phi)}{\mathcal{I}_{D/2-1}(\kappa_\phi)} - \frac{\mathcal{I}_{D/2}(\kappa_\phi)(\mathcal{I}_{D/2-2}(\kappa_\phi)\mathcal{I}_{D/2}(\kappa_\phi))}{\mathcal{I}_{D/2-1}(\kappa_\phi)^2}\right) + 1\right), \end{aligned} \tag{23}$$

where the chain rule is applied to derive the gradient of the variational parameter ϕ. In Equation (23), κ_ϕ represents the function that computes the concentration parameter κ of the variational vMF distribution, which is the encoder network in our model, and the gradient with respective to ϕ, i.e., $\nabla_\phi \kappa_\phi$ can be handled by automatic differentiation packages. When combining with the expression of the gradient of KL divergence with respective to the concentration parameter κ_ϕ, we can obtain the gradient of the variational parameter ϕ. Similar to Davidson et al. [9], we use the exponentially scaled modified Bessel function to replace the modified Bessel functions for numerical stability.

5.5 Sampling and Reparameterization

To optimize the objective function, we use the **Stochastic Gradient Variational Bayes** (**SGVB**) algorithm and the reparameterization trick proposed by Kingma and Welling [28], which requires us to sample from the latent distribution and compute the stochastic gradient of the variational parameter ϕ using reparameterization. In this section, we first describe an acceptance-rejection sampling technique to get samples from the latent von Mises–Fisher distribution and then compute gradient with respect to variational parameter using reparameterization under the acceptance-rejection sampling scheme.

Sampling. The algorithm for sampling from a von Mises–Fisher distribution is summarized in Algorithm 1. In order to sample from a von Mises–Fisher distribution $q(\mathbf{z}|\boldsymbol{\mu}, \kappa)$, we follow the sampling procedure in [58], where we first sample from a von Mises–Fisher distribution $q(\mathbf{z}'|\boldsymbol{\mu}_1, \kappa)$ with mean direction vector $\boldsymbol{\mu}_1 = (1, 0, \ldots, 0)$. During sampling from $q(\mathbf{z}'|\boldsymbol{\mu}_1, \kappa)$, we need to first sample ω, which is distributed as $g(\omega|\kappa, D) \propto \exp(\omega\kappa)(1 - \omega^2)^{\frac{1}{2}(D-3)}$, using acceptance-rejection sampling and the procedure is given in part I of Algorithm 1, where $\mathrm{Beta}(\alpha_1, \alpha_2)$ and $\mathrm{Uniform}(\beta_1, \beta_2)$ denote a Beta distribution with parameters α_1 and α_2, and a uniform distribution with parameters β_1 and β_2, respectively. After that, we sample vector $\mathbf{v}$ from the uniform distribution defined on the $(D-2)$ dimensional sphere $U(\mathcal{S}^{D-2})$. By combining ω and $\mathbf{v}$ we can obtain sample $\mathbf{z}' \sim q(\mathbf{z}'|\boldsymbol{\mu}_1, \kappa)$, which is shown in part II of Algorithm 1, where $\mathrm{Concat}(\cdot, \cdot)$ means the

ALGORITHM 1: Sampling of the von Mises–Fisher distribution.

```
   Input  : Mean direction vector $\boldsymbol{\mu}$
            Concentration parameter $\kappa$
            Dimension of the samples $D$
   Output : Sample $\mathbf{z}$ of the von Mises–Fisher distribution $q(\mathbf{z}|\boldsymbol{\mu},\kappa)$
 1 /* Part I: Sample $\omega \sim g(\omega|\kappa, D) \propto \exp(\omega\kappa)(1-\omega^2)^{\frac{1}{2}(D-3)}$ by acceptance-rejection sampling,
      which can be completed by steps 2 to 11.                                  */
 2 Initialize values:
 3     $b \leftarrow \frac{-2\kappa+\sqrt{4\kappa^2+(D-1)^2}}{D-1}$
 4     $a \leftarrow \frac{(D-1)+2\kappa+\sqrt{4\kappa^2+(D-1)^2}}{4}$
 5     $d \leftarrow \frac{4ab}{1+b} - (D-1)\ln(D-1)$
 6 repeat
 7     Sample $\epsilon \sim \text{Beta}(\frac{1}{2}(D-1), \frac{1}{2}(D-1))$
 8     $\omega \leftarrow h(\epsilon,\kappa) = \frac{1-(1+b)\epsilon}{1-(1-b)\epsilon}$
 9     $t \leftarrow \frac{2ab}{1-(1-b)\epsilon}$
10     sample $u \sim \text{Uniform}(0,1)$
11 until $(D-1)\ln(t) - t + d \geq \ln(u)$
12 /* Part II: Sample $\mathbf{z}'$ of the von Mises--Fisher distribution $q(\mathbf{z}'|\boldsymbol{\mu}_1,\kappa)$ with
      $\boldsymbol{\mu}_1 = (1,0,\ldots,0)$, which can be completed by steps 13 to 14.      */
13 Sample vector $\mathbf{v} \sim U(\mathcal{S}^{D-2})$
14 Obtain sample $\mathbf{z}' \leftarrow \text{Concat}(\omega; \sqrt{1-\omega^2}\mathbf{v})$
15 /* Part III: Obtain sample $\mathbf{z}$ of the von Mises--Fisher distribution $q(\mathbf{z}|\boldsymbol{\mu},\kappa)$ by
      householder transformation, which can be completed by steps 16 to 19.     */
16 $\mathbf{u}' \leftarrow \boldsymbol{\mu}_1 - \boldsymbol{\mu}$
17 $\mathbf{u} \leftarrow \frac{\mathbf{u}'}{||\mathbf{u}'||}$
18 $\mathbf{W} \leftarrow \mathbb{I} - 2\mathbf{u}\mathbf{u}^\top$
19 $\mathbf{z} \leftarrow \mathbf{W}\mathbf{z}'$
20 Return $\mathbf{z}$
```

concatenation operation. In part III of Algorithm 1, after obtaining sample $\mathbf{z}'$, we can make it distributed as $q(\mathbf{z}|\boldsymbol{\mu},\kappa)$ by using the householder transformation which is an orthogonal transformation such that $\mathbf{W}\boldsymbol{\mu}_1 = \boldsymbol{\mu}$.

Reparameterization. In classic VAEs, it is straight-forward to compute the gradient of the generative parameter θ, but the gradient with respect to the variational parameter ϕ can be problematic. In order to compute the gradient of variational parameter ϕ with samples from the latent distribution, Kingma and Welling [28] propose a reparameterization trick that introduces an auxiliary variable ϵ and expresses the latent variable $\mathbf{z}$ as a deterministic function $\mathbf{z} = h_\phi(\epsilon, \mathbf{x})$ with $\mathbf{x}$ being the input and h_ϕ representing a differentiable function parameterized by ϕ. Recently, Naesseth et al. [46] have introduced a technique that makes it possible to extend the reparameterization trick to distributions using an acceptance-rejection scheme to sample data, which we utilize in our algorithm as follows. In the case of sampling from a von Mises–Fisher distribution, we need to sample ω from $g(\omega|\kappa_\phi, D)$, where we use $g(\omega|\phi)$ for simplicity, by accepting or rejecting samples from the proposal distribution $r(\omega|\phi)$. After obtaining samples of ω, we can express the latent variable $\mathbf{z}$ as $\mathbf{z} = t(\omega;\phi)$ with t as a differentiable function. We introduce an auxiliary variable $\epsilon \sim s(\epsilon)$, which is a Beta distribution in our case, and let $\omega = h(\epsilon, \phi)$ with h being a differentiable function with respect to ϕ. Details of the $s(\epsilon)$ and h can be found in Algorithm 1.

For notational simplicity, we represent the expectation terms in the objective function of Equation (17) as $\mathbb{E}_{q_\phi}[f(\mathbf{z})] = \mathbb{E}_{g(\omega|\phi)}[f(t(\omega;\phi))]$ with f being a certain function of $\mathbf{z}$. Using the proposition proven in [46], the expectation term can be represented as follows:

$$\mathbb{E}_{q_\phi}[f(\mathbf{z})] = \mathbb{E}_{g(\omega|\phi)}[f(t(\omega;\phi))] \tag{24}$$
$$= \mathbb{E}_{\pi(\epsilon;\phi)}[f(t(h(\epsilon,\phi);\phi))], \tag{25}$$

with $\pi(\epsilon;\phi) = s(\epsilon)\frac{g(h(\epsilon,\phi)|\phi)}{r(h(\epsilon,\phi)|\phi)}$ being the distribution of the accepted samples ϵ. It is given by the rejection sampling, where we sample a value $\epsilon \sim s(\epsilon)$ and uniform value $u \sim \text{Uniform}(0,1)$ and accept the sample if $u < \frac{g(h(\epsilon,\phi)|\phi)}{M_\phi r(h(\epsilon,\phi)|\phi)}$. Then we can compute the gradient of the expected term with respect to the variational parameter ϕ using the log derivative trick:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f(\mathbf{z})] \\ &= \nabla_\phi \mathbb{E}_{\pi(\epsilon;\phi)}[f(t(h(\epsilon,\phi);\phi))] \\ &= \mathbb{E}_{\pi(\epsilon;\phi)}[\nabla_\phi f(t(h(\epsilon,\phi);\phi))] + \mathbb{E}_{\pi(\epsilon;\phi)}\left[f(t(h(\epsilon,\phi);\phi))\nabla_\phi \log \frac{g(h(\epsilon,\phi)|\phi)}{r(h(\varepsilon,\phi)|\phi)}\right].\end{aligned} \tag{26}$$

In Equation (26) above, the first term is the reparameterization term as in [28], while the second term is the correction term that accounts for not using $r(\omega|\phi) \equiv q(\omega|\phi)$ for sampling ω. Thus, we can obtain an unbiased Monte Carlo estimate of the gradient of the reconstruction loss with respect to the variational parameter ϕ by acceptance-rejection sampling from $\pi(\varepsilon;\phi)$.

5.6 Optimization

To enable scalable optimization, the standard VAE [28] replaces the inference of latent variables with an inference network and the generative process with a generative network. The models are trained by learning the parameters of the inference and generative networks. Similarly, to effectively optimize our model, we introduce an inference network f_ϕ with variational parameters ϕ and a generative network g_θ with generative parameters θ, which serve as the encoder and the decoder of the proposed model, respectively. Both the variational parameter ϕ and the generative parameter θ are trained jointly by maximizing the objective function (i.e., Equation (17)) using the reparameterization trick described above. We show the details of our model in this section.

Inference network. As mentioned before, we aim at embedding nodes and attributes of the attributed network in a hyperspherical space and this can be achieved by mapping the features of nodes and attributes, i.e., $\mathbf{F}^{\mathcal{V}}$ and $\mathbf{F}^{\mathcal{A}}$, to latent von Mises–Fisher distributions using the inference network f_ϕ. There are two types of inference networks for embedding nodes and attributes, respectively.

To infer the latent von Mises–Fisher distributions of nodes, we apply a two-layer **Graph Convolutional Network (GCN)** [30] by taking both the adjacency matrix and the attributes matrix as input and output the mean vector $\boldsymbol{\mu}_{\phi_1}$ and the concentration parameter κ_{ϕ_1} of the latent von Mises–Fisher distribution. We use a GCN instead of other neural networks for its representational learning ability and useful learning biases

$$\mathbf{H}_{\mathcal{V}} = \text{ReLU}\left(\tilde{\mathbf{A}}\mathbf{F}^{\mathcal{V}}\mathbf{W}_{\mathcal{V}}^{(0)}\right), \tag{27}$$
$$\boldsymbol{\mu}_{\mathcal{V}} = \text{Norm}\left(\tilde{\mathbf{A}}\mathbf{H}_{\mathcal{V}}\mathbf{W}_{\mathcal{V}}^{(1)}\right), \tag{28}$$
$$\kappa_{\mathcal{V}} = \text{Softplus}\left(\tilde{\mathbf{A}}\mathbf{H}_{\mathcal{V}}\mathbf{W}_{\mathcal{V}}^{(2)}\right), \tag{29}$$

where $\mathbf{H}_{\mathcal{V}}$ represents the hidden layer of the inference network of nodes, and $\boldsymbol{\mu}_{\mathcal{V}}, \kappa_{\mathcal{V}}$ represent the mean vector and the concentration parameter of the learned vMF distributions over embeddings of nodes, respectively. $\text{ReLU}(x) = \max(0, x)$ and $\text{Softplus}(x) = \log(1 + \exp(x))$ are the non-linear activation functions, while $\text{Norm}(\cdot)$ is the normalization operation that forces the direction vector $\boldsymbol{\mu}_{\phi_1}$ to be a unit vector. Besides, $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ representing the degree matrix of the graph $\mathcal{G}$. $\phi_1 = \{\mathbf{W}_{\mathcal{V}}^{(0)}, \mathbf{W}_{\mathcal{V}}^{(1)}, \mathbf{W}_{\mathcal{V}}^{(2)}\}$ are the variational parameters of the node inference network.

As for the attributes, we infer von Mises–Fisher distributions using a two-layered fully connected neural network, which is given by:

$$\mathbf{H}_{\mathcal{A}} = \text{ReLU}\left(\mathbf{F}^{\mathcal{A}}\mathbf{W}_{\mathcal{A}}^{(0)} + \mathbf{b}^{(0)}\right), \tag{30}$$

$$\boldsymbol{\mu}_{\mathcal{A}} = \text{Norm}\left(\mathbf{H}_{\mathcal{A}}\mathbf{W}_{\mathcal{A}}^{(1)} + \mathbf{b}^{(1)}\right), \tag{31}$$

$$\kappa_{\mathcal{A}} = \text{Softplus}\left(\mathbf{H}_{\mathcal{A}}\mathbf{W}_{\mathcal{A}}^{(2)} + \mathbf{b}^{(2)}\right), \tag{32}$$

where $\boldsymbol{\mu}_{\mathcal{A}}$ and $\kappa_{\mathcal{A}}$ are the parameters of the von Mises–Fisher distributions over embeddings of attributes. $\mathbf{b} = \{\mathbf{b}^{(0)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}\}$ is the bias of the fully connected layer and $\phi_2 = \{\mathbf{W}_{\mathcal{A}}^{(0)}, \mathbf{W}_{\mathcal{A}}^{(1)}, \mathbf{W}_{\mathcal{A}}^{(2)}\}$ are the trainable parameters of the attribute inference network.

The overall variational parameters of the inference networks are denoted as $\phi = \{\phi_1, \phi_2\}$. After optimization, the variational von Mises–Fisher distributions over nodes and attributes can be inferred through the variational networks. Using the sampling technique introduced in Section 5.5 to sample from the von Mises–Fisher distributions, we can obtain the latent representations $\mathbf{Z}^{\mathcal{V}}$ and $\mathbf{Z}^{\mathcal{A}}$ of nodes and attributes, which are used in the generative network to reconstruct the observations, while allowing for computing gradients of the variational parameters using the reparameterization trick.

Generative network. The goal of the generative network (i.e., the decoder) of the proposed model is to reconstruct the observations, i.e., the adjacency matrix $\mathbf{A}$ and attribute matrix $\mathbf{X}$, using the embeddings of the nodes $\mathbf{Z}^{\mathcal{V}}$ and attributes $\mathbf{Z}^{\mathcal{A}}$. We describe the generative process of the edge in the adjacency matrix and the attribute in the node attribute matrix as follows:

(i) For each node $\mathcal{V}_i$ and attribute $\mathcal{A}_a$, draw latent embeddings:

$$\mathbf{Z}_i^{\mathcal{V}} \sim \text{vMF}(\boldsymbol{\mu}_{\mathcal{V},i}, \kappa_{\mathcal{V},i}), \tag{33}$$

$$\mathbf{Z}_a^{\mathcal{A}} \sim \text{vMF}(\boldsymbol{\mu}_{\mathcal{A},a}, \kappa_{\mathcal{A},a}). \tag{34}$$

(ii) We start by describing the generative process of the adjacency matrix. We represent the decoder network of edges as g_{θ_1}, which takes embeddings of nodes $\mathbf{Z}_i^{\mathcal{V}}$, $\mathbf{Z}_j^{\mathcal{V}}$ as input and outputs the parameters for the corresponding distribution of the edge:

$$\left[\boldsymbol{\mu}_{\theta_1}, \sigma_{\theta_1}^2\right] = g_{\theta_1}\left(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}\right). \tag{35}$$

For each edge $\mathbf{A}_{ij}$ in the adjacency matrix $\mathbf{A}$:

(a) If edge $\mathbf{A}_{ij}$ is real-valued, then:

$$p_{\theta_1}\left(\mathbf{A}_{ij}|\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}\right) = \mathcal{N}\left(\mathbf{A}_{ij}|\boldsymbol{\mu}_{\theta_1}, \sigma_{\theta_1}^2\mathbf{I}\right). \tag{36}$$

(b) If edge $\mathbf{A}_{ij}$ is binary, then:

$$p_{\theta_1}\left(\mathbf{A}_{ij}|\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}\right) = \text{Ber}(\mathbf{A}_{ij}|\boldsymbol{\mu}_{\theta_1}), \tag{37}$$

where $\mathcal{N}(\mathbf{A}_{ij}|\boldsymbol{\mu}_{\theta_1}, \sigma^2_{\theta_1})$ and $\text{Ber}(\mathbf{A}_{ij}|\boldsymbol{\mu}_{\theta_1})$ are the Gaussian distribution and Bernoulli distribution of the edge $\mathbf{A}_{ij}$ parameterized by $\boldsymbol{\mu}_{\theta_1}, \sigma^2_{\theta_1}\mathbf{I}$ and $\boldsymbol{\mu}_{\theta_1}$, respectively.

(iii) We denote the generative network of the attribute matrix $\mathbf{X}$ as g_{θ_2}, which takes embeddings of nodes $\mathbf{Z}_i^{\mathcal{V}}$ and attributes $\mathbf{Z}_a^{\mathcal{A}}$ as input and outputs parameters for the distribution of the attribute value:

$$\left[\boldsymbol{\mu}_{\theta_2}, \sigma^2_{\theta_2}\right] = g_{\theta_2}\left(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}\right). \tag{38}$$

For each attribute value $\mathbf{X}_{ia}$ in the attribute matrix $\mathbf{X}$:

(a) If $\mathbf{X}_{ia}$ is real-valued, then:

$$p_{\theta_2}\left(\mathbf{X}_{ia}|\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}\right) = \mathcal{N}\left(\mathbf{X}_{ia}|\boldsymbol{\mu}_{\theta_2}, \sigma^2_{\theta_2}\mathbf{I}\right). \tag{39}$$

(b) If $\mathbf{X}_{ia}$ is binary, then:

$$p_{\theta_2}\left(\mathbf{X}_{ia}|\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}\right) = \text{Ber}(\mathbf{X}_{ia}|\boldsymbol{\mu}_{\theta_2}). \tag{40}$$

Similarly, $\mathcal{N}(\mathbf{X}_{ia}|\boldsymbol{\mu}_{\theta_2}, \sigma^2_{\theta_2})$ and $\text{Ber}(\mathbf{X}_{ia}|\boldsymbol{\mu}_{\theta_2})$ are the Gaussian distribution and Bernoulli distribution, respectively, whose parameters are the output of the attribute decoder network g_{θ_2} based on the embeddings of node $\mathcal{V}_i$ and attribute $\mathcal{A}_a$.

We denote all the parameters of the decoder network as $\theta = \{\theta_1, \theta_2\}$. Since all the edges and the attributes in our experiments are binary-valued, we implement our generative model simply by the inner product between the latent representations, which is given by

$$g_{\theta_1}\left(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}\right) = \text{Sigmoid}\left(\left\langle \mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}\right\rangle\right), \tag{41}$$

$$g_{\theta_2}\left(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}\right) = \text{Sigmoid}\left(\left\langle \mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}\right\rangle\right), \tag{42}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors and Sigmoid$(\cdot)$ is the sigmoid function. We use simple inner product as our generative model because it produces the best results in all experiments including node classification, link prediction and attribute inference.

Time complexity analysis. As we can see from the above discussions, the main computational cost of the HCAN model comes from the layer-wise propagations for the node encoder network and the attribute encoder network. The time complexity of the two-layer GCN network of node encoder for each epoch is $C_1 = O(N(NF + FH_1 + NH_1 + H_1D))$, where N and F are the number of nodes and attributes in the network, respectively, H_1 is the dimension of the first GCN layer and D is the dimension of embeddings. For the two-layer neural network of attribute encoder, the time complexity for each epoch is $C_2 = O(FH_2(N + D))$, where H_2 is the dimension of the first hidden layer. Hence, the overall time complexity of our co-embedding model HCAN is $C = C_1 + C_2$. Empirically, our HCAN costs around 15s per 10 epoch on the Cora dataset during training for the task of link prediction on an Intel i7 3.60 GHz CPU computer.

5.7 Comparison with Other Network Embedding Methods

To obtain a better understanding of HCAN, we provide explicit comparisons between HCAN and other network embedding methods. Firstly, network embedding approaches based on the random walks or edge sampling, such as DeepWalk [48], node2vec [18], and LINE [56], are the most widely studied methods. They use random walks on the network to generate context pairs and use the skip-gram loss to learn embeddings while HCAN is based on a variational auto-encoder and utilizes the reconstruction errors of the adjacency matrix and attribute matrix to learn embeddings.

Other approaches based on VAEs have also been proposed, such as the VGAE [29] and S-VGAE [9]. However, these methods learn embeddings for nodes only and it is unclear how to obtain embeddings of attributes in the network, while our co-embedding model can jointly learn embeddings for both nodes and attributes in a unified hyperspherical space and capture the similarities between them.

In another line of work, some approaches, which are based on graph convolution networks for learning embeddings of nodes in attributed networks, have also been proposed, such as the GraphSAGE [20], the **graph attention network** (**GAT**) [20], and the **deep graph infomax** (**DGI**) [60]. However, these methods learn point vectors as the network embeddings, which are unable to capture uncertainties in the networks. In contrast, HCAN learns von Mises–Fisher distributions as the network embeddings and the variances of these distributions can capture uncertainties in the networks.

6 EXPERIMENTAL SETUP

In this section, we detail our experimental setup. We first list our research questions in Section 6.1 and then describe the experimental datasets in Section 6.2. Baselines are given in Section 6.3, and training and parameter settings are provided in Section 6.4.

6.1 Research Questions

The research questions guiding the remainder of the article are as follows:

RQ1 How does the proposed model HCAN perform on node-oriented network tasks, e.g., link prediction and node classification?

RQ2 Can the HCAN model that embeds nodes and attributes in a unified hyperspherical space effectively capture the similarities between nodes and attributes of the network? How does it perform on the attribute inference task?

RQ3 Can we qualitatively evaluate the learned embeddings of the nodes and attributes in HCAN?

RQ4 Is HCAN sensitive to the hyperparameters, such as the free parameter α in Equation (17) and the embedding size D, in link prediction and attribute inference tasks?

RQ5 What are the effects of the major components such as the hyperspherical embeddings, the uncertainty estimation and the co-embedding mechanism in HCAN?

6.2 Datasets

We conduct experiments on eight real-world attributed network datasets with their statistics showing in Table 2:

Cora, Citeseer, and Pubmed [53]: These three datasets are citation networks where the nodes are papers and attributes are the words in those papers. The edges of the network are citation links, while the labels are the topics of the papers.

BlogCatalog [57]: BlogCatalog is a social relationship network of bloggers from the BlogCatalog website. The nodes represent the users while the attributes are the keywords of the blogs generated by the users. The labels represent the topic categories given by the authors.

Facebook [32]: This dataset is built by SNAP[1] using the profiles and the relationship data of 10 users in Facebook, where users are treated as nodes, while the attributes are generated by their profiles.

DBLP: This dataset is crawled from the DBLP public bibliography data,[2] from which we construct a collaboration network. We treat each author as a node in the network with the collaboration between two authors as the edge. We also extract top 172 computer science conferences

[1]Available from: http://snap.stanford.edu/data/.
[2]Available from: http://dblp.uni-trier.de/xml/.

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
Facebook	4,039	88,234	1,406	–
DBLP	12,213	131,713	172	–
Physics	34,493	247,962	8,415	5
Ogbl-citation	2,927,963	30,561,187	–	–

(Tier A and B conferences according to the China Computer Federation[3]) as attributes of the nodes.

Physics [54]: The Physics dataset is a co-authorship network where nodes represent authors, that are connected by an edge if they co-authored a paper. The node attributes represent paper keywords for each author's papers, which are real-valued, and class labels indicate most active fields of study for each author.

Ogbl-citation [23]: This is a large-scale citation network collected for link prediction where nodes represent papers, edges are citations between nodes, and each node is associated with a 128-dimensional WORD2VEC [45] feature vector. Since there are no explicit attributes and labels in the dataset, we only use it for the link prediction task.

6.3 Baselines

We list the baseline models that we consider for comparison in this section. To evaluate the performance of HCAN on link prediction and node classification tasks (**RQ1**), we compare it against nine state-of-the-art attributed network embedding methods:

AANE [24]: This is an attributed network embedding model that learns embeddings of nodes based on the decomposition of node attribute proximity. We use the same hyperparameter settings as the original paper. For a fair comparison, we set the dimension of embeddings to be 20, which is the same as for HCAN.

GraphSAGE [20]: GraphSAGE learns embeddings of nodes in the network by sampling and aggregating features from nodes' local neighborhoods. GraphSAGE has different variants based on different feature aggregators, and we adopt GraphSAGE with mean-based aggregator as our baseline. In our experiments, we use a two-layer GraphSAGE network where the dimensions of the hidden layer and output layer are 64 and 20, respectively. Moreover, we randomly sample 20 and 10 neighbors for aggregation in the hidden layer and the output layer, respectively.

ANRL-WAN [72]: The model learns embeddings for attributed network by using the neighbor enhancement autoencoder to model the node attribute information and using attribute-aware skip-gram model to capture network structure. We adopt one of its variants that uses the Weighted Average Neighbor function to construct its target neighbors, abbreviated as ANRL-WAN. For the architecture of the auto-encoder, we use two-layer neural network with hidden dimension of 64 and output dimension of 20. In the skip-gram loss, we

[3]Available from: http://www.ccf.org.cn/.

use random walks with a walk length of 80 and window size of 10 to generate the context pairs.

VGAE [29]: VGAE learns latent Gaussian embeddings for attributed network by using the variational auto-encoder, which is trained by maximizing the ELBO of the likelihood of the adjacency matrix. It only embeds nodes in Euclidean space instead of both nodes and attributes like HCAN. In VGAE, we use two-layer graph neural network as the encoder, where the hidden dimension is 64 and output dimension is 20. Similar to that in HCAN, we use the inner product between embeddings as the decoder.

GAE [29]: GAE is similar to VAGE, where the Gaussian embeddings are replaced with vector embeddings, and is optimized by minimizing the reconstruction loss of the adjacency matrix. We use a similar network architecture as VAGE in our experiments.

Node2vec [18]: Node2vec learns embeddings of nodes based on random walks over the network structure of data. Similar to that in ANRL-WAN, we use random walks with a walk length of 80 and window size of 10 to generate the context pairs, and use the skip-gram loss to learn embeddings with a dimension of 20.

S-VGAE [9]: S-VGAE is a generalization of VGAE, where the Gaussian distribution is replaced with the von Mises–Fisher distribution such that it obtains hyperspherical embeddings. However, it only learns embeddings for nodes instead of both nodes and attributes like our HCAN. We also use a similar network architecture as VGAE in our experiments.

GMNN [49]: GMNN employs a conditional random field to model the joint distribution of object labels, and uses two GCNs for the inference and learning procedures. In our experiments, we use the unsupervised version of GMNN where the neighbors of each node are treated as the labels. The dimension of hidden layers of the GCNs is set to 64 and the dimension of the embeddings is 20.

DGI [60]: DGI learns node embeddings by maximizing the mutual information between representations of local subgraphs and the high-level representation of the graph, both of which are obtained through the GCN. In our experiments, we use a two-layer GCN with 64-dimensional hidden layer and 20-dimensional output layer to learn representations.

To answer **RQ2**, we compare HCAN against seven baselines in terms of attribute inference task:

SAN [71]: The model performs joint link prediction and attribute inference based on a random walks with restart algorithm on the attribute-augmented social network. We use the same random walk length as Node2vec and use a restart probability of 0.3.

EdgeExp [4]: This is an attribute inference algorithm that leverages a softmax function to solve for both user attributes and relationship types. The dimension of the embeddings is 20 and the number of neighbors is set to 100.

BLA [69]: This is a probabilistic model that iteratively learns user links and attributes, and leverages data redundancy on each side and mutual reinforcement between the two. We also use 20-dimensional embeddings and set the β parameter, which is a trade-off between edge existence and transition probability, to 0.5.

LRA-SAN [15]: This method learns embeddings of both nodes and attributes by the low-rank approximation of the attribute-augmented adjacency matrix, which is the concatenation of attribute matrix and adjacency matrix. We use 20-dimensional embeddings.

CN-SAN [15]: This is an extension of SAN that additionally assigns a score for a node-attribute pair based on the features of their common neighbors.

AA-SAN [15]: This is also an extension of SAN that additionally calculates the Adamic-Adar scores based on the node degrees.

NGCF [64]: NGCF is a recommendation algorithm that designs an embedding propagation layer for leveraging collaborative signals to learn embeddings. In our experiments, we treat nodes as users and the attributes as items to employ NGCF. We use a two-layer embedding propagation layer where the dimensions of the hidden and output layer are 64 and 20, respectively.

6.4 Settings

As mentioned in Section 5.1, the architecture of HCAN contains two main parts: the inference model and the generative model. We adopt a two-layer GCN neural network [30] and a two-layer fully-connected neural network for the node inference model and the attribute inference model, respectively. In all experiments, the hidden layers of the two inference networks are set to 512, while the dimension of the latent embeddings is 20 unless specifically stated. As for the generative network, we simply use the inner product of the learned embeddings to reconstruct the adjacency matrix and attribute matrix as it yields best performance in all our experiments. We implement HCAN using PyTorch [47].[4]

HCAN is trained by maximizing the objective function in Equation (17) and we optimize it using the Adam optimizer [26] with a learning rate of 0.03. In each experiment, we train HCAN for 250 iterations to obtain the resulting embeddings. The parameter α that balances the weight of the reconstruction accuracy of the adjacency matrix and attribute matrix, is chosen within the range $[0.1, 0.9]$ and is tuned to obtain the best performance for the tasks of link prediction, attribute inference and node classification. Specifically, since the Ogbl-citation dataset is too large for a GCN to train in a GPU, we replace the two-layer GCN in the node inference network with a two-layer GraphSAGE network with GCN aggregator [20], which enables scalable mini-batch training. In this case, we use a batch size of 512, a learning rate of 0.0005, and train the model for $1,500$ iterations. Similarly, we also replace the GCNs of baselines with the GraphSAGE network on the Ogbl-citation dataset. For the baseline models, we use the code released by the authors and use the hyperparameter settings in Section 6.3. The input features of both HCAN and the baselines are normalized as unit length vectors, i.e., unit normalized directional vectors, with a preprocessing step.

7 RESULTS

In this section, we report our experimental results. We address **RQ1** by evaluating the proposed model on two graph mining tasks, i.e., link prediction and node classification. We then utilize the learned embeddings of both nodes and attributes of the graph on an attribute inference task to answer **RQ2**. Then we address **RQ3** by visualizing both node and attribute embeddings of a DBLP network in 3-D space. Subsequently, we answer **RQ4** by varying the free parameter α and the embedding size D in link prediction and attribute inference tasks. Finally, we conduct ablation studies to examine the effects of the major components in HCAN to address **RQ5**.

7.1 Link Prediction and Node Classification

The goal of link prediction is to predict if there exists an edge between two nodes and it is a typical task in social network analysis. We first compare the performance of HCAN on the link prediction task with the baselines. Specifically, we follow the experimental settings in [29, 30], where we randomly divide all the edges in the network into training set (85%), validation set (5%), and test set (10%). We also randomly sample an equal number of non-existing edges as negative samples in all three sets. After the model is optimized, we simply take the value of the inner product of the

[4]The code of HCAN is publicly available from https://github.com/fangjy6/HCAN.

Table 3. Mean Performance of Link Prediction of HCAN and the Baseline Models

Method	Cora		Citeseer		Pubmed		Facebook		BlogCatalog		Physics		Ogbl-citation	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
AANE	.755	.753	.821	.841	.778	.765	.843	.834	.698	.711	.798	.784	-	-
GraphSAGE	.803	.819	.815	.804	.879	.847	.857	.846	.729	.701	.943	.930	.910	.912
ANRL-WAN	.835	.851	.843	.846	.911	.889	.935	.916	.772	.751	.849	.833	-	-
VAGE	.929	.931	.908	.923	.935	.929	.971	.964	.808	.804	.935	.927	.954	.958
GAE	.927	.933	.899	.918	.931	.930	.964	.957	.803	.806	.933	.920	.947	.950
Node2vec	.813	.832	.796	.825	.877	.859	.788	.801	.671	.691	.871	.866	.844	.857
S-VAGE	.942	.940	.945	.952	.940	.934	.969	.968	.810	.812	.931	.924	.961	.965
GMNN	.902	.889	.918	923	.925	.911	.955	.947	.793	.795	.867	.855	-	-
DGI	.944	.939	.948	.957	.929	.917	.948	.933	768	754	.893	.901	-	-
HCAN	**.973**†	**.968**†	**.987**†	**.984**†	**.954**†	**.948**†	**.983**†	**.975**†	**.829**†	**.823**†	**.954**†	**.945**†	**.979**†	**.977**†

The best results per metric per dataset are marked in boldface. In each dataset, significant improvements over the comparative methods are marked with † (paired t -test, p < .05).

learned embeddings of two nodes as the probability of a link between them. We use area under the ROC curve (AUC) and **average precision** (**AP**) score [29] as our evaluation metrics to evaluate the performance of HCAN on the link prediction task.

Table 3 shows the link prediction results of HCAN and the baseline models. Note that some experimental results of baselines on Ogbl-citation dataset are missing due to the out of memory issue. From the experimental results, we obtain the following observations. (i) HCAN significantly outperforms all the baselines on all datasets according to the paired t-test ($p < .05$). Specifically, HCAN achieves higher than 95% AUC (Area under the ROC Curve) and AP (Average Precision) scores on the Cora, Citeseer, and Facebook datasets, and the improvements on both Citeseer and Physics are significant, which demonstrates that the latent embeddings of nodes learned by HCAN are effective for the link prediction task. (ii) Both S-VAGE and HCAN, which learn hyperspherical embeddings, can outperform GAE and VAGE, which learn Gaussian embeddings. This is because the hyperspherical embeddings can better preserve the spherical nature of the input features. The results show that it is more effective to learn hyperspherical embeddings than hyperplane embeddings when the input features are unit-normalized. (iii) Compared with S-VAGE, GAE, and VGAE, which embed nodes only, HCAN, which co-embeds nodes and attributes jointly, achieves better performance. This is because these baselines only try to reconstruct the adjacency matrix and ignore the reconstruction of attribute matrix, which leads to the less effective embeddings. Our experimental results share similar findings of previous work [4, 42, 71], which demonstrates that leveraging attribute information in addition to the network structure can improve the performance of link prediction. (iv) The GCN-based methods, i.e., GraphSAGE, VGAE, DGI, and HCAN, can significantly outperform the other methods, i.e., AANE, ANRL-WAN, and Node2vec, since GCNs can effectively aggregate multi-hop neighborhood information to learn useful representations, which is difficult for methods based on random walks or neural networks. The results suggest the effectiveness of utilizing GCN as inference network in HCAN.

To further examine the usefulness of the learned node embeddings, we apply the embeddings in a downstream node classification task, which is a classical task for evaluating the quality of embeddings. Similar to the experimental settings in [24], after obtaining the latent embeddings of nodes, we randomly sample 20% of the nodes as labelled nodes to train the logistic regression classifier and randomly select 1,000 nodes in the remainder for evaluation. To evaluate the performance of the classifier, we use Macro_F1 and Micro_F1 as evaluation metrics [24, 72]. We repeat the experiment for 10 times and report the average performance on both Macro_F1 (Ma_F1) and

Table 4. Mean Node Classification Performance of HCAN and the Baseline Models

Method	Cora		Citeseer		Pubmed		BlogCatalog		Physics	
	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1
AANE	.636	.623	.589	.647	.804	.798	.603	.613	.621	.697
GraphSAGE	.734	.758	.598	.639	.797	.808	.617	.638	.889	.919
ANRL-WAN	.774	.741	**.671**	**.718**	.764	.773	$\underline{.609}$	.621	.843	.879
VAGE	.742	.766	.544	.586	.805	.812	.558	.579	.897	$\underline{.920}$
GAE	.745	.762	.468	.536	.790	.802	.547	.561	.872	.912
Node2vec	.584	.544	.443	.515	.698	.707	.448	.476	.757	.791
S-VAGE	$\underline{.779}$	.802	.589	.672	.804	.813	.554	.592	.894	.916
GMNN	.784	$\underline{.810}$	.624	.649	$\underline{.808}$	$\underline{.815}$	.607	$\underline{.628}$	$\underline{.901}$	.917
DGI	.625	.701	.604	.687	.798	.812	.514	.605	.883	.903
HCAN	**.808**†	**.827**†	$\underline{.656}$	$\underline{.708}$	**.816**†	**.823**†	**.623**†	**.655**†	**.911**†	**.937**†

The best and the second best results per metric per dataset are marked in boldface and underlined, respectively. For each dataset, significant improvements over the comparative methods are marked with † (paired t-test, p < .05).

Micro_F1 (Mi_F1). Table 4 shows the classification performance of our model and the baselines. Note that we do not use the Facebook and Ogbl-citation datasets for the classification task because there is no label information in these two datasets.

As shown in Table 4, HCAN achieves the best performance on four out of five datasets (i.e., Cora, Pubmed, BlogCatalog, and Physics) and achieves second-best performance on the Citeseer dataset compared with state-of-the-art baselines. The results show that HCAN can learn useful and effective embeddings for the downstream node classification task. Specifically, when compared with the VGAE model, HCAN consistently and significantly achieves a higher performance (paired t-test, $p < .05$) on all five datasets, and HCAN can obtain a 8.9% and 8.0% absolute increase in terms of the Ma_F1 and Mi_F1 metrics on the Cora dataset, which confirms the effectiveness of the hyperspherical embeddings in the node classification task. Moreover, HCAN consistently outperforms S-VAGE on all five datasets, which verifies that the co-embedding model is more effective in the node classification task. Consequently, by combining the experimental results in both link prediction and node classification, we can conclude that HCAN is more effective in learning high-quality embeddings of nodes than baselines for downstream node-oriented network tasks such as link prediction and node classification.

Furthermore, to qualitatively show that HCAN can lean high-quality hyperspherical embeddings, we visualize the learned latent embeddings of HCAN on the Cora dataset, which is shown in Figure 3. In the figure, we also visualize the embeddings obtained by the competitive baseline models, e.g., AANE, ANRL, and VAGE, for comparison. Specifically, after we obtain the leaned embeddings of nodes for each method, we use the t-SNE tool [40] to map them into the corresponding low-dimensions. It is shown that HCAN can obtain better, more compact and separated cluster results compared with the baseline models, which helps to explain why HCAN can achieve better classification results on the Cora dataset.

7.2 Attribute Inference

Next, we answer **RQ2** by examining the performance of HCAN on attribute inference tasks compared with baselines. The target of attribute inference is to predict the value of attributes of nodes in the network, and hence capturing and measuring similarities between nodes and attributes in this task is important. Since most baselines in the link prediction task, such as AANE, GraphSAGE, S-VAGE, GMNN, DGI, and so on, only learn node embeddings and hence are unable to capture similarities between nodes and attributes, they cannot be applied in attribute inference task.

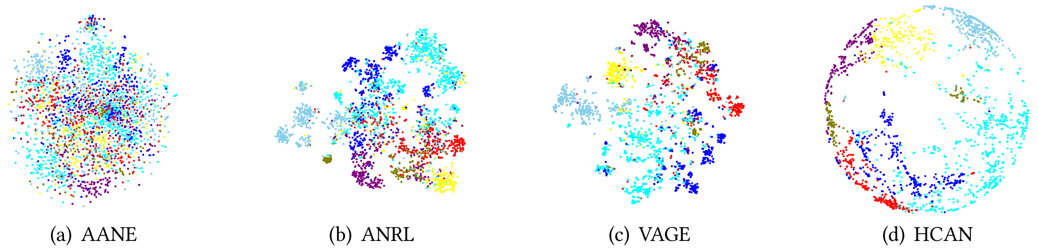


Fig. 3. Visualization of the latent space of the inferred embeddings on the Cora citation network for HCAN and baselines. The same color indicates the same class label, which are not provided during training.

For comparison, we take seven state-of-the-art attribute inference algorithms, i.e., SAN, LRA-SAN, CN-SAN, AA-SAN, EdgeExp, BLA, and NGCF, as our baselines. We adopt the same experimental setup as in the link prediction task, where we randomly split attributes of nodes into three sub-sets: training (85%), validating (5%), and testing (10%) sets, and randomly sample an equal number of non-existing node-attribute pairs in all three sets. For evaluation metrics, we employ the AUC and AP metrics to evaluate the attribute inference performance.

Table 5 shows the attribute inference results of HCAN and the baseline models on six attributed networks. We find that HCAN performs significantly better than all the baseline methods on all datasets (paired t-test, $p < .05$). The NGCF method can obtain the second best performance in most cases. The reason why HCAN and NGCF obtain a better performance than other baselines is that they both optimize a loss function containing the reconstruction error of all the attributes. Moreover, compared with NGCF, HCAN not only optimizes the reconstruction error of all the attributes but also optimizes a reconstruction error of the network structure. The fact that HCAN achieves better performance than NGCF suggests that it is beneficial to also consider the network structure in addition to the attribute matrix for improving the performance in the attribute inference task.

Our experimental findings are consistent with the principle of social influence [31], which states that users who are linked are likely to adopt similar attributes and suggests that network structure should inform the attribute inference task. In addition, it is worth noting that all the baselines have relatively poor performance on the Pubmed dataset, while HCAN obtains a markedly better performance. Attribute inference on the Pubmed dataset is challenging because the attribute matrix in the Pubmed dataset is quite sparse; the ratio between the positive node-attribute pairs and the negative node-attribute pairs is only 0.02. The sparsity of the attribute matrix results in a challenging, highly unbalanced classification problem. The experimental results in Table 5 show that HCAN is capable of learning effective latent hyperspherical embeddings for attribute inference task by capturing the similarities between nodes and attributes in the network.

7.3 Network Visualization

Next we turn to **RQ3** to qualitatively evaluate the learned embeddings of HCAN by visualizing the embeddings of both nodes and attributes on the DBLP network in a unified hyperspherical space. The DBLP network is an academic social network where the nodes are authors and the attributes are the conferences at which an author has published academic publications. We choose DBLP as experimental dataset for better interpretation of the similarities between nodes and attributes. Specifically, we co-embed the DBLP academic network to obtain 3-dimensional embeddings for the authors and the conferences. Then we plot the embeddings on 3-dimensional hyperspherical

Table 5. Mean Attribute Inference Performance of HCAN and the Baselines

Method	**Cora**		**Citeseer**		**Pubmed**		**Facebook**		**BlogCatalog**		**Physics**	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
EdgeExp	.685	.701	.709	.729	.599	.586	.686	.697	.684	.749	.858	.854
SAN	.664	.681	.685	.697	.579	.587	.712	.723	.694	.720	.794	.809
BLA	.798	.784	.812	.795	.625	.619	.868	.831	.787	.792	.866	.870
LRA-SAN	.665	.668	.690	.677	.586	.594	.730	.722	.669	.685	.812	.827
CN-SAN	.725	.704	.755	.741	.618	.601	.798	.804	.731	.744	.845	.844
AA-SAN	.751	.747	.805	.809	.699	.690	.851	.844	.801	.807	.884	.891
NGCF	.793	.787	.822	.803	.725	.692	.901	.907	.812	.810	.930	.933
HCAN	**.810**†	**.796**†	**.831**†	**.818**†	**.761**†	**.736**†	**.915**†	**.917**†	**.824**†	**.825**†	**.944**†	**.939**†

The best performance per metric per dataset are marked in boldface. In each dataset, significant improvements over the comparative methods are marked with † (paired t -test, $p < .05$).

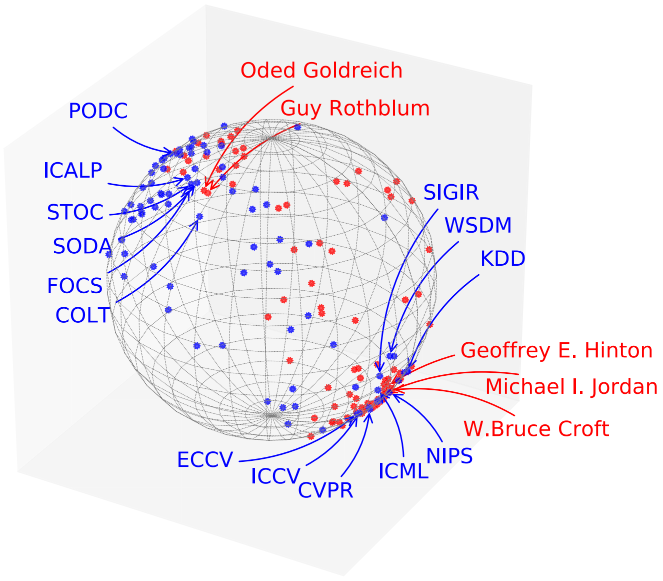


Fig. 4. The 3-dimensional visualization of hyperspherical embeddings of authors and conferences for the DBLP dataset. The red nodes are the top 200 high H-index authors, while the blue nodes are the 172 conferences in our DBLP dataset.

space. For visual clarity, we only plot the embeddings of 200 authors who have the top 200 high H-index[5] on the dataset. The visualization results are shown in Figure 4, from which we have the following observations:

(i) The embeddings of similar conferences are displayed quite closely to each other at the same region on the hypersphere. Specifically, the five conferences with a heavy emphasis on data-driven methods and machine learning, i.e., ICML, KDD, NIPS and SIGIR, and CVPR, are shown at the bottom right region of the hypersphere. In contrast, the five theoretical computer science conferences, i.e., FOCS, ICALP, PODC, SODA, and STOC are plotted at the top

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

left region of the hypersphere. This result again indicates that HCAN is capable of capturing meaningful similarities between attributes of the network.

(ii) The embeddings of similar scholars are also shown to be close on the latent space. As shown in the figure, experts in the field of machine learning, information retrieval and data mining, e.g., Michael I. Jordan, Geoffrey E. Hinton, and W. Bruce Croft, are quite close at the bottom right region while Oded Goldreich and Guy Rothblum, who are experts in theoretical computer science, are shown at the top left region of the sphere. This finding illustrates that HCAN is also able to capture the similarities between nodes of the network.

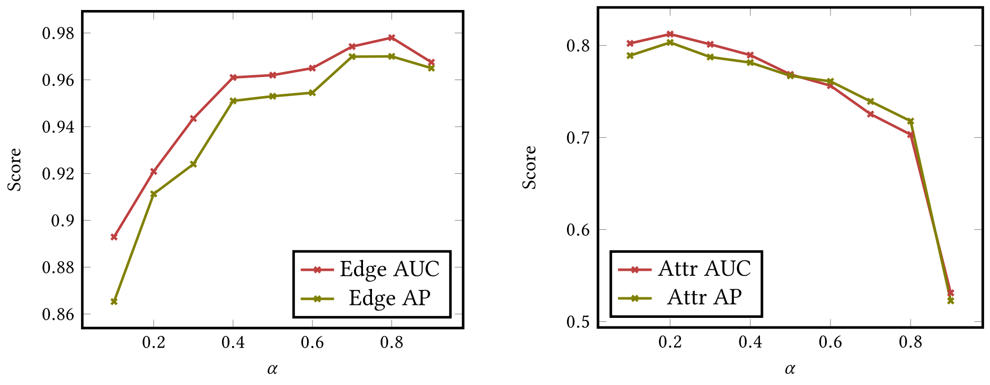
(iii) Finally, the visualization also shows that experts in the field of machine learning, e.g., Michael I. Jordan, are quite close to conferences in the same field, e.g., NIPS, while experts in theoretical computer science, e.g., Oded Goldreich, are quite close to conferences in theoretical computer science, e.g., SODA. The result confirms that HCAN can effectively capture similarities between nodes and attributes of the network.

7.4 Hyperparameter Sensitivity

Next, we turn to **RQ4** to understand whether the performance of HCAN is sensitive to the hyperparameters such as the free parameter α and the embedding size D. Specifically, we examine the performance of HCAN with different values of α and D in the link prediction and attribute inference tasks.

7.4.1 The Effect of Free Parameter α. In the HCAN method, a hyperparameter α is introduced to balance the reconstruction accuracy between adjacency matrix and attribute matrix, which performs a trade-off between link prediction task and attribute inference task. To understand the effect of this parameter α, we conduct an experiment on the Cora dataset: we calculate the AUC and AP score of link prediction and attribute inference under different settings of α, which ranges from 0.1 to 0.9, and the result is shown in Figure 5. According to the figure, as α increases from 0.1 to 0.9, the performance on the link prediction task improves, while the performance on attribute inference gets worse. This is quite intuitive, as α governs how important the reconstruction of the adjacency matrix is in our model: when α increases, HCAN weights more on the reconstruction error of adjacency matrix and hence obtains better performance on link prediction task and worse performance on attribute inference task. The result indicates that we are able to optimize for the link prediction or attribute inference tasks by the tuning parameter α and thus obtain a task-specific model. Moreover, the figure shows that the performance of link prediction increases rapidly as α increases from 0.1 to 0.5 and then gets smoother when α is greater than 0.5, which suggests the performance of link prediction is more sensitive to α when α is smaller than 0.5. In contrast, the performance of attribute inference decreases rapidly when α is greater than 0.5, which suggests that the performance of attribute inference is more sensitive to α when α is greater than 0.5. The sensitivity analysis helps to better fine-tune the parameter α to obtain the best performance on link prediction and attribute inference.

7.4.2 The Effect of the Embedding Size. Next, we examine the effects of the embedding size on the performance of HCAN in the tasks of link prediction and attribute inference. To study the effect of the embedding size, we vary the size of latent embeddings of HCAN, i.e., D, from 5 to 50 and report the performance of learned embeddings in both link prediction and attribute inference tasks on three citation networks, i.e., Cora, Citeseer, and Pubmed. We only report experimental results on the citation networks because we found similar results on the other datasets. We repeat each experiment for 10 times and report the mean and the standard deviation of the link prediction and attribute inference performance evaluation scores (AUC and AP). Moreover, we also compare HCAN against DGI on the link prediction task. Specifically, we report the link



(a) Link Predition (b) Attribute Inference

Fig. 5. Link prediction and attribute inference performance of HCAN on the Cora dataset by varying the smoothing parameter α.

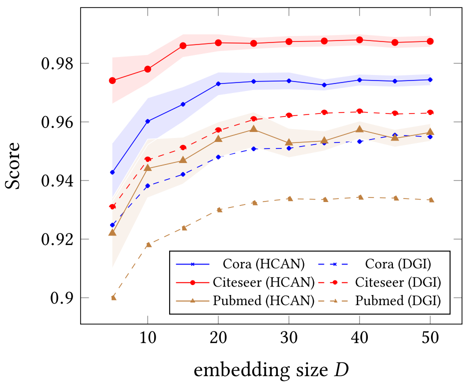
prediction performance of DIG with different embedding sizes. We only report the results of DGI since DGI has the best performance on the link prediction compared with other baselines. Similarly, we compare HCAN against NGCF on the attribute inference task. The experimental results are presented in Figure 6. Note that we only report the mean evaluation scores of the representative baselines, i.e., DGI and NGCF, for clarity.

As shown in the figure, in both the link prediction and attribute inference tasks, the performance of our HCAN on all three datasets increases rapidly when the embedding size increases from 5 to 20, and then gradually converges when the embedding size is greater than 20. The results suggest that HCAN is not very sensitive to the embedding size once it is greater than a certain threshold. This can be explained based on the fact that our unit-length hyperspherical embeddings only record the orientations of objects on the hypersphere and it is sufficient to use a small embedding size to achieve that purpose. The findings show another merit of HCAN: we can use a small embedding size in HCAN to achieve satisfactory performance in both link prediction and attribute inference tasks. Another result shown in Figure 6 is that as the embedding size increases from 5 to 50, the standard deviations of evaluation metrics (AUC and AP) on link prediction and attribute inference tasks gradually decrease. This suggests that HCAN learns more stable and robust object embeddings when using a larger embedding size.

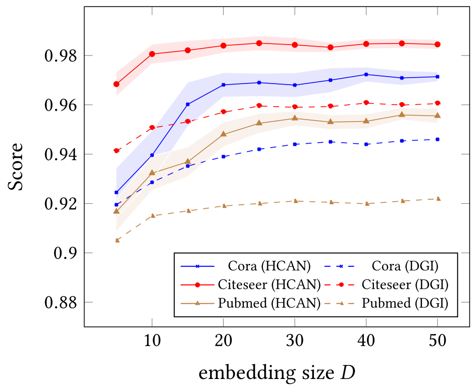
Moreover, the results in Figure 6 show that the performance of DGI on the link prediction task gradually converges when the embedding size is greater than 20, which indicates that DGI is also not very sensitive to the embedding size, and it is sufficient to use a small embedding size to obtain satisfactory performance. Similarly, the performance of NGCF in attribute inference also shows the insensitivity to embedding size. Similar results can also be found for the remaining baselines. These experimental results validate that both HCAN and the baselines are insensitive to the embedding size and simply increasing the embedding size cannot significantly improve the performance. As a result, it is fine to compare HCAN and the baselines using the same embedding size. Moreover, Figure 6 also shows that, under the same embedding size, HCAN can always outperform the baselines on both the link prediction and attribute inference tasks, which again confirms the effectiveness of the proposed HCAN in learning useful network embeddings.

7.5 Ablation Studies

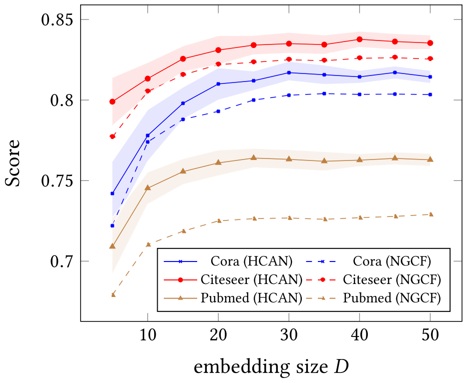
Finally, we conduct ablation studies to examine the effect of the major components in HCAN to address **RQ5**. Specifically, we study the effect of hyperspherical embeddings, the effect of the



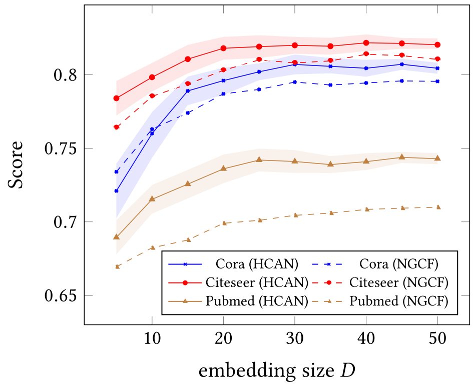
(a) Performance (AUC) of Link Predition



(b) Performance (AP) of Link Prediction



(c) Performance (AUC) of Attribute Inference



(d) Performance (AP) of Attribute Inference

Fig. 6. Link prediction and attribute inference performance of HCAN on the Cora, Citeseer, and Pubmed datasets by varying the embeddings size D of our HCAN. The lines represent the mean of evaluation metrics, while the shadow areas represent the standard deviation of evaluation metrics.

uncertainty estimation, and the effect of the co-embedding method, so as to validate the utility and effectiveness of each of these components in our HCAN.

7.5.1 The Effect of Hyperspherical Embeddings. Firstly, we examine the effectiveness of hyperspherical embeddings by comparing against Gaussian embeddings. It is straightforward to employ Gaussian embeddings in the HCAN framework by simply replacing the von Mises–Fisher distributions of node embeddings and attribute embeddings with Gaussian distributions and using the reparameterization trick [28] for optimization. For a fair comparison, the dimension of the Gaussian embeddings is the same as the hyperspherical embeddings, i.e., $D = 20$. Table 6 shows the performance of link prediction and attribute inference of HCAN under different types of embeddings on six attributed networks. As shown in the figure, HCAN with hyperspherical embeddings outperforms HCAN with Gaussian embeddings in both the link prediction and attribute inference tasks on all datasets. This is because the input features of the attributed networks are unit normalized in the preprocessing step, which removes the magnitude of the features and only keeps the orientation of features as discriminative spherical information. Hence, the hyperplane Gaussian embeddings in a Euclidean space may not be the appropriate representations concerning the spherical features of data, which can explain the suboptimal performance

Table 6. Link Prediction and Attribute Inference Performance of HCAN when Using Gaussian Embeddings and Hyperspherical Embeddings

Task	Method	Cora		Citeseer		Pubmed		Facebook		BlogCatalog		Physics	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Link prediction	**Gaussian**	.951	.946	.958	.954	.920	.922	.936	.935	.739	.738	.921	.914
	VMF	.973	.968	.987	.984	.954	.948	.983	.975	.829	.823	.954	.945
Attribute inference	**Gaussian**	.803	.801	.821	813	.754	.740	.901	.905	.815	.817	.937	.931
	VMF	.810	.796	.831	.818	.761	.736	.915	.917	.824	.825	.944	.939

Note that "Gaussian" represents using Gaussian distribution as embeddings and "VMF" represents using von Mises-Fisher distribution as embeddings.

Table 7. Link Prediction and Attribute Inference Performance of HCAN under Different Settings of Whether or Not to Estimate the Uncertainties of Embeddings

Task	Method	Cora		Citeseer		Pubmed		Facebook		BlogCatalog		Physics	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Link Prediction	**without κ**	.965	.960	.978	.979	.946	.933	.979	.969	.806	.788	.946	.940
	with κ	.973	.968	.987	.984	.954	.948	.983	.975	.829	.823	.954	.945
Attribute Inference	**without κ**	.789	.788	.823	.809	.753	.729	.908	.899	.818	.796	.938	.931
	with κ	.810	.796	.831	.818	.761	.736	.915	.917	.824	.825	.944	.939

Note that "with κ" represents to estimate the uncertainties, while "without κ" represents not to estimate the uncertainties.

of Gaussian embeddings. The experimental results suggest that hyperspherical representations are more effective than the hyperplane representations for data with spherical features.

7.5.2 The Effect of Uncertainty Estimation. Secondly, we study the effect of uncertainties κ in learned hyperspherical embeddings. In HCAN, we use the von Mises–Fisher distributions as latent embeddings of objects in the attributed network, where the variances of the distributions, i.e., κ, can capture the uncertainties of embeddings. To validate the effectiveness of this uncertainty estimation, we conduct ablation studies of HCAN by removing the variance. In such case, the embeddings of objects reduce to unit-length deterministic vectors. The experimental results in Table 7 show that HCAN with the uncertainty estimation ("with κ") can slightly outperform HCAN without the uncertainty estimation ("without κ") in both link prediction and attribute inference tasks on all the datasets. This is due to the fact that most real-world datasets are inherently noisy due to the data collection and data preparation processes, where errors commonly occur. By estimating the uncertainties of embeddings, we allow some tolerance for these types of noise and learn more robust representations from the noisy data, which can explain the superior performance of HCAN with the uncertainty estimation. The experimental results validate the effectiveness of the uncertainty estimation in our HCAN and suggests that HCAN is able to learn useful and robust embeddings of objects in the noisy attributed networks.

7.5.3 The Effect of the Co-embedding Method. Thirdly, to examine the effect of the co-embedding mechanism, we compare the embeddings of nodes and attributes learned with the co-embedding model HCAN against the embeddings of nodes and attributes separately learned from the adjacency matrix and attribute matrix. Specifically, we use the same node inference network and decoder network as HCAN for learning hyperspherical embeddings of nodes by minimizing the reconstruction loss of the adjacency matrix. Similarly, we also learn the embeddings of attributes by minimizing the reconstruction loss of the attribute matrix. We then compare the

Table 8. Link Prediction and Attribute Inference Performance of Embeddings Learned with Our Co-embedding Model HCAN and the Embeddings Separately Learned from the Adjacency Matrix and the Attribute Matrix

Task	**Method**	**Cora**		**Citeseer**		**Pubmed**		**Facebook**		**BlogCatalog**		**Physics**	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Link Prediction	**Seperate**	.961	.956	.979	.980	.948	.941	.973	.969	.799	.791	.942	.930
	Co-embedding	.973	.968	.987	.984	.954	.948	.983	.975	.829	.823	.954	.945
Attribute Inference	**Seperate**	.795	.791	.804	.805	.753	.731	.907	.901	.816	.820	.934	.929
	Co-embedding	.810	.796	.831	.818	.761	.736	.915	.917	.824	.825	.944	.939

performance of learned embeddings in both link prediction and attribute inference tasks, where the results are represented in Table 8. As shown in the table, our co-embedding model HCAN achieves better performance on both link prediction and attribute inference tasks on all the datasets. The reason that HCAN performs better on the link prediction task is that the attribute information can inform the interactions between nodes, since two nodes with similar attributes are more likely to link to one another. Moreover, the network structure can help infer missing attribute values of nodes, since nodes that are linked together are more likely to adopt similar attributes, which can explain the superior performance of HCAN on the attribute inference task.

The experimental results confirm the effectiveness of our co-embedding model for jointly learning the embeddings of nodes and attributes. Additionally, previous studies [15, 43, 44] share the same empirical findings as ours, which suggests that jointly considering the network structure and node-attribute information can improve the performance of link prediction and attribute inference.

8 CONCLUSION

In this article, we have studied the problem of learning representations for attributed networks with unit normalized directional features by modeling the embeddings with a non-Gaussian and non-Euclidean representation space. We have shown that variational auto-encoders can be effectively generalized to heterogeneous data with multiple types of entities and relations, and proposed a hyperspherical variational co-embedding model, called HCAN, to map the nodes and attributes of an attributed network into a unified hyperspherical representation space such that the similarities between them can be effectively captured and measured. Based on the framework of VAEs, HCAN contains inference networks that map the input entities to latent hyperspherical embeddings and generative networks that reconstruct the input based on the embeddings. We have detailed the derivation of the objective function used in HCAN and have designed practical proper trainable inference and generative networks to enable scalable optimization of the objective.

We have conducted extensive experiments on six real-world attributed networks. The experimental results show that HCAN outperforms the baselines on several different application tasks, e.g., link prediction and attribute inference, which indicates that HCAN is able to learn high-quality hyperspherical embeddings for both nodes and attributes of attributed networks while being able to capture the similarities between them. The 3-Dimensional visualization of the DBLP dataset indicates that the similarities between nodes and attributes can be properly preserved and measured by HCAN by mapping them in a unified hyperspherical space. Moreover, by introducing a free parameter to balance the reconstruction accuracy between adjacency matrix and attribute matrix, we can obtain a task-specific model.

We would like to discuss some broader implications of our work. In the information retrieval literature, it is critical to leverage network-based information to boost the performance of many traditional retrieval tasks as much real-world data is often represented by attributed networks, such as the purchase networks, social networks, and hyper-linked web documents. Our method can be easily applied to solve down-streaming problems based on these network data by learning useful and effective network embeddings. For instance, it is straightforward to apply our method to solve the recommendation problem in user/item purchase networks, the user profiling problem in social networks, and document classification problem in the hyper-linked website data. Moreover, this article reveals that jointly learning embeddings of different categories of entities in the data is more effective than learning their embeddings separately. Our co-embedding framework might inspire researchers to develop more effective and useful embedding techniques in other research domains.

We would also like to discuss some limitations of our model. Since HCAN is an unsupervised method that only uses node features and geometric structures for learning, one potential limitation is that our model may not directly leverage other important properties of attributed networks, such as the label information and the community structures. Additionally, the proposed generalized variational auto-encoder framework can also be applied to heterogeneous data with multiple types of entities and relations. However, in this article, we mainly focus on its performance on attributed networks, and its performance on heterogeneous data remains unexplored, which we leave as to further work.

As to future work, we intend to apply HCAN to other tasks for which representations of two or more different categories of entities need to be inferred in the same semantic space, such as retrieval-based question answering, for which questions and answers need to be co-embedded, or document retrieval, for which queries and documents need to be co-embedded, or entity retrieval, for which queries and entities need to be co-embedded. In addition, we plan to propose a dynamic co-embedding algorithm for dynamic attributed networks, where the evolution of the embeddings of both nodes and attributes over time should be properly modeled.

ACKNOWLEDGMENTS

We thank the reviewers for their helpful and constructive suggestions.

REFERENCES

[1] Caglar Aytekin, Xingyang Ni, Francesco Cricri, and Emre Aksu. 2018. Clustering and unsupervised anomaly detection with l 2 normalized deep auto-encoder representations. In *Proceedings of the 2018 International Joint Conference on Neural Networks*. IEEE, 1–6.

[2] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *Proceedings of the International Conference on Learning Representations*.

[3] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*. 10–21.

[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 31th International Conference on Machine Learning*, Vol. 32. 874–882.

[5] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. 2020. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*. 1–44.

[6] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2019. Joint neural collaborative filtering for recommender systems. *ACM Transactions on Information Systems* 37, 4 (2019), 1–30.

[7] Yong Cheng. 2019. Semi-supervised learning for neural machine translation. In *Proceedings of the Joint Training for Neural Machine Translation*. Springer, 25–40.

[8] Taco Cohen, Mario Geiger, Jonas Köhler, and Max Welling. 2017. Convolutional networks for spherical signals. *arXiv preprint arXiv:1709.04893*. 1–5.

[9] Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. 2018. Hyperspherical variational auto-encoders. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI'18)*. 856–865.

[10] Inderjit S. Dhillon and Suvrit Sra. 2003. *Modeling Data using Directional Distributions*. Technical Report TR-03-06. Department of Computer Sciences, The University of Texas at Austin.

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

[12] Jinyuan Fang, Shangsong Liang, Zaiqiao Meng, and Qiang Zhang. 2021. Gaussian process with graph convolutional kernel for relational learning. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1–11.

[13] Nicholas I. Fisher, Toby Lewis, and Brian J. J. Embleton. 1993. *Statistical Analysis of Spherical Data*. Cambridge University Press.

[14] Hongchang Gao and Heng Huang. 2018. Deep attributed network embedding. In *Proceedings of the IJCAI*, Vol. 18. 3364–3370.

[15] 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 Transactions on Intelligent Systems and Technology* 5, 2 (2014), 27.

[16] Siddharth Gopal and Yiming Yang. 2014. Von mises-fisher clustering models. In *Proceedings of the International Conference on Machine Learning*. 154–162.

[17] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37. 1462–1471.

[18] 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.

[19] Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics* 6 (2018), 437–450.

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

[21] Md Hasnat, Julien Bohné, Jonathan Milgram, Stéphane Gentric, and Liming Chen. 2017. Von mises-fisher mixture model-based deep learning: Application to face verification. *arXiv preprint arXiv:1706.04264*. 1–16.

[22] 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 Conference on Information and Knowledge Management*. ACM, 623–632.

[23] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of the Advances in Neural Information Processing Systems*.

[24] 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.

[25] 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.

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

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

[28] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*.

[29] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. In *Proceedings of the NIPS Workshop on Bayesian Deep Learning*.

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

[31] Timothy La Fond and Jennifer Neville. 2010. Randomization tests for distinguishing social influence and homophily effects. In *Proceedings of the 19th International Conference on World Wide Web*. 601–610.

[32] Jure Leskovec and Julian J. Mcauley. 2012. Learning to discover social circles in ego networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 539–547.

[33] 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.

[34] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*. 689–698.

[35] Shangsong Liang, Yupeng Luo, and Zaiqiao Meng. 2021. Profiling users for question answering communities via flow-based constrained co-embedding model. In *Proceedings of the ACM Transactions on Information Systems* (2021), 1–38.

[36] Shangsong Liang, Emine Yilmaz, and Evangelos Kanoulas. 2016. Dynamic clustering of streaming short documents. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 995–1004.

[37] Shangsong Liang, Emine Yilmaz, and Evangelos Kanoulas. 2018. Collaboratively tracking interests for user clustering in streams of short texts. *IEEE Transactions on Knowledge and Data Engineering* 31, Issue 2 (2018), 257–272.

[38] 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 and Data Mining*. ACM, 1764–1773.

[39] Yi Liao, Wai Lam, Lidong Bing, and Xin Shen. 2018. Joint modeling of participant influence and latent topics for recommendation in event-based social networks. *ACM Transactions on Information Systems* 36, 3 (2018), 1–31.

[40] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008), 2579–2605.

[41] Kantilal Varichand Mardia. 2014. *Statistics of Directional Data*. Academic press.

[42] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding attributed networks. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. ACM, 393–401.

[43] Zaiqiao Meng, Shangsong Liang, Jinyuan Fang, and Teng Xiao. 2019. Semi-supervisedly co-embedding attributed networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 6507–6516.

[44] Zaiqiao Meng, Shangsong Liang, Xiangliang Zhang, Richard McCreadie, and Iadh Ounis. 2020. Jointly learning representations of nodes and attributes for attributed networks. *ACM Transactions on Information Systems* 38, 2 (2020), 1–32.

[45] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems*. 3111–3119.

[46] Christian A. Naesseth, Francisco J. R. Ruiz, Scott W. Linderman, and David M. Blei. 2017. Reparameterization gradients through acceptance-rejection sampling algorithms. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, Vol. 54. 489–498.

[47] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Proceedings of the NIPS-W*.

[48] 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.

[49] Meng Qu, Yoshua Bengio, and Jian Tang. 2019. GMNN: Graph markov neural networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 5241–5250.

[50] Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2020. Knowledge graphs: An information retrieval perspective. *Foundations and Trends in Information Retrieval* 14, 4 (2020), 1–158.

[51] Joseph Reisinger, Austin Waters, Bryan Silverthorn, and Raymond J. Mooney. 2010. Spherical topic models. In *Proceedings of the 27th International Conference on Machine Learning*. 903–910.

[52] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, Vol. 32. 1278–1286.

[53] 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), 93–93.

[54] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868. 1–11.

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

[56] 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.

[57] 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.

[58] Gary Ulrich. 1984. Computer generation of distributions on the M-Sphere. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 33, 2 (1984), 158–163.

[59] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*.

[60] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2018. Deep graph infomax. In *Proceedings of the International Conference on Learning Representations*.

[61] Pengfei Wang, Hanxiong Chen, Yadong Zhu, Huawei Shen, and Yongfeng Zhang. 2019. Unified collaborative filtering over graph embeddings. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 155–164.

[62] Weiqing Wang, Hongzhi Yin, Xingzhong Du, Wen Hua, Yongjun Li, and Quoc Viet Hung Nguyen. 2019. Online user representation learning across heterogeneous social networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 545–554.

[63] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *Proceedings of the AAAI*. 203–209.

[64] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–174.

[65] Libing Wu, Cong Quan, Chenliang Li, Qian Wang, Bolong Zheng, and Xiangyang Luo. 2019. A context-aware user-item representation learning for item recommendation. *ACM Transactions on Information Systems* 37, 2 (2019), 1–29.

[66] Teng Xiao, Shangsong Liang, Weizhou Shen, and Zaiqiao Meng. 2019. Bayesian deep collaborative matrix factorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5474–5481.

[67] Binbin Xu, Sarthak Pathak, Hiromitsu Fujii, Atsushi Yamashita, and Hajime Asama. 2017. Spatio-temporal video completion in spherical image sequences. *IEEE Robotics and Automation Letters* 2, 4 (2017), 2032–2039.

[68] Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4503–4513.

[69] 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.

[70] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7370–7377.

[71] Zhijun Yin, Manish Gupta, Tim Weninger, and Jiawei Han. 2010. A unified framework for link recommendation using random walks. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 152–159.

[72] 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 *Proceedings of the IJCAI*. 3155–3161.

[73] Wayne Xin Zhao, Yupeng Hou, Junhua Chen, Jonathan J. H. Zhu, Eddy Jing Yin, Hanting Su, and Ji-Rong Wen. 2020. Learning semantic representations from directed social links to tag microblog users at scale. *ACM Transactions on Information Systems* 38, 2 (2020), 1–30.

Received August 2020; revised July 2021; accepted July 2021