Deep Variational Network Embedding in Wasserstein Space

Dingyuan Zhu*
Tsinghua University
zhudy11@126.com

Peng Cui
Tsinghua University
cuip@tsinghua.edu.cn

Daixin Wang
Tsinghua University
dxwang0826@gmail.com

Wenwu Zhu
Tsinghua University
wwzhu@tsinghua.edu.cn

ABSTRACT

Network embedding, aiming to embed a network into a low dimensional vector space while preserving the inherent structural properties of the network, has attracted considerable attentions recently. Most of the existing embedding methods embed nodes as point vectors in a low-dimensional continuous space. In this way, the formation of the edge is deterministic and only determined by the positions of the nodes. However, the formation and evolution of real-world networks are full of uncertainties, which makes these methods not optimal. To address the problem, we propose a novel Deep Variational Network Embedding in Wasserstein Space (**DVNE**) in this paper. The proposed method learns a Gaussian distribution in the Wasserstein space as the latent representation of each node, which can simultaneously preserve the network structure and model the uncertainty of nodes. Specifically, we use 2-Wasserstein distance as the similarity measure between the distributions, which can well preserve the transitivity in the network with a linear computational cost. Moreover, our method implies the mathematical relevance of mean and variance by the deep variational model, which can well capture the position of the node by the mean vectors and the uncertainties of nodes by the variance. Additionally, our method captures both the local and global network structure by preserving the first-order and second-order proximity in the network. Our experimental results demonstrate that our method can effectively model the uncertainty of nodes in networks, and show a substantial gain on real-world applications such as link prediction and multi-label classification compared with the state-of-the-art methods.

KEYWORDS

Network Embedding, Wasserstein space, Deep Learning

ACM Reference Format:
Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. 2018. Deep Variational Network Embedding in Wasserstein Space. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220052

1 INTRODUCTION

Network embedding has attracted considerable research attentions in the past few years. The basic idea is to embed a network into a low-dimensional vector space to preserve the network structure. Many network embedding methods are demonstrated to be effective in a variety of applications, such as link prediction [42, 44], classification [8, 26] and clustering [35, 46]. However, most of existing network embedding methods represent each node by a single point in a low-dimensional vector space. In this way, the formation of the whole network structure is deterministic.

Actually, real-world networks are much more complex than we assume. The formation and evolution of the networks are full of uncertainties. For example, for the nodes with low degree, they contain less information and thus their representations bear more uncertainties than others. For the nodes across multiple communities, the possible contradiction between their neighboring nodes may also be larger and thus cause the uncertainty. Furthermore, in social network, human behavior is multi-faceted which also makes the generation of edges uncertain [47]. For all of these cases, without considering the uncertainty of networks, the learned embeddings will be less effective in network analysis and inference tasks.

Gaussian distribution innately represents the uncertainty property [43]. Therefore, it is promising to represent a node by Gaussian distributions, i.e. the mean and the variance, rather than a point vector to incorporate the uncertainty. Motivated by this, to model the uncertainty of each node using Gaussian distributions, there are some basic requirements for network embedding methods to meet.

- **Transitivity**: The embedding space should be a metric space to preserve the transitivity in networks. Transitivity is a very important property in networks, especially in social networks [25]. For example, the friend of my friend is more likely to be my friend than some randomly chosen users. Moreover, the transitivity measures the density of triangles in a network, which plays an important role in calculating clustering coefficient [7]. If the metric space satisfies the triangle inequality, the transitivity in the network can be well preserved.
- **Uncertainty**: By using Gaussian distributions to represent a node, the mean and the variance should preserve different

[*] Beijing National Research Center for Information Science and Technology(BNRist)

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220052

properties to make such representations informative. Specifically, the mean vectors should reflect the position of the nodes and variance terms should contain the uncertainty of the nodes. In this way, the representations based on distributions can preserve the uncertainty while supporting network applications.

- **Strcutural Proximity**: The network structures, especially high-order proximity, should be preserved in a effective and efficient way. The high-order proximity is critical for capturing the network structure, which has been demonstrated to be useful in many real-world applications [36].

Recently, some works attempt to use Gaussian distributions to represent a node for network embedding [3, 17, 24] to integrate uncertainty. However, these methods use the Kullback-Leibler (abbreviated as KL) divergence [28] to measure the similarity between distributions. However, the KL divergence is asymmetric and does not satisfy triangle inequality. Thus, it can not well preserve the transitivity of proximity in networks, especially in undirected networks. Additionally, these methods regard the variance terms as additional dimensions of mean vectors, and use similarity measure to constrain their learning. In this way, they do not reflect the intrinsic relationship between variance terms and mean vectors in the model. Finally, very few of these works preserve the high-order proximity in network embedding, except Graph2Gauss [3]. But Graph2Gauss needs to calculate the shortest path between any two nodes, which is unaffordable in large-scale networks.

To address these problems, we propose a novel Deep Variational Network Embedding in Wasserstein Space method in this paper, named **DVNE**. The proposed method learns a Gaussian Embedding for each node in the Wasserstein Space by the deep variational model. Specifically, we employ 2-Wasserstein distance to measure the similarity between the distributions, i.e. the embeddings of the nodes. The 2-Wasserstein distance is a real metric that able to preserve the transitivity in embedding space. In this way, the proposed deep model is able to simultaneously preserve the transitivity and model the node uncertainty with linear time complexity. Meanwhile, we use a deep variational model to minimize the Wasserstein distance between the model distribution and the data distribution, which can extract the intrinsic relationship between mean vectors and variance terms. Furthermore, our method efficiently preserve the first-order and second-order proximity of the nodes in networks, empowering the learned node representations to reflect both local and global network structure [44].

The main contributions of our method are summarized as follows:

- We propose DVNE, an novel method that learns the Gaussian embedding in the Wasserstein space, which can well preserve the transitivity in networks and reflect the uncertainties of nodes.
- We imply the mathematical relevance of mean vectors and variance terms by the deep variational model, where the mean vectors denote the position of the nodes and the variance terms represent the uncertainties of the nodes.
- We efficiently preserve the first-order and second-order proximity between nodes, thus the learned representations capture the local and global network structure.
- We comprehensively evaluate the effectiveness of DVNE on several real-world networks in various applications.

The rest of the paper is organized as follows. In Section 2, we review the related work. In Section 3, we summarize the notations used in this paper and give the problem formulation. We introduce the framework of the method in Section 4 and report the experimental results in Section 5. We conclude the paper in Section 6.

2 RELATED WORK

Because of the popularity of networked data, network embedding has received more and more attentions in recent years. We briefly review some network embedding methods, and readers can referred to [13] for a comprehensive survey. Deepwalk [37] first uses the language modeling technique to learn the latent representations of a network by truncated random walks. LINE [39] embeds the network into a low-dimensional space where the first-order and second-order proximity between nodes are preserved. Node2vec [22] learns a mapping of nodes to a low-dimensional space of features that maximizes the likelihood of preserving network neighborhoods of nodes. HOPE [36] proposes a high-order proximity preserved embedding method. Furthermore, deep learning method for network embedding is also studied. SDNE [44] first considers the high nonlinearity in network embedding and proposes a deep autoencoder to preserve the first- and the second-order proximities. The graph variational autoencoder (GAE) [27] learns node embeddings in an unsupervised manner with variational autoencoder (VAE) [16].

All the aforementioned methods learn a point-vector for each node as its embedding. However, as we stated before, these methods have the limitation to model the uncertainty, which is a critical property needed to be considered for network embedding. Then some following works start to consider the uncertainty problem. Inspired by [43], which learns the Gaussian word embeddings to model uncertainty, KG2E [24] learns Gaussian embeddings for knowledge graphs. HCGE [17] similarly learns Gaussian embeddings for heterogeneous graphs. And Aleksandar et al. [3] proposes a deep model to learn Gaussian embeddings on the attributed network. All of these methods use the KL divergence or its variant JensenShannon divergence [19] as the similarity measure between the distributions. However, both the KL divergence and the JensenShannon divergence are not the true metrics. These metrics do not satisfy the triangle inequality. In this way, these methods cannot preserve the transitivity to get effective representations for networks. Furthermore, these methods regard the variance terms as the extra dimensions, then use the similarity measure to constrain their learning. In this way, it is difficult to capture the intrinsic relationships between the mean and the variance terms.

3 NOTATIONS AND PROBLEM DEFINITION

In this section, we summarize the notations used in this paper and give the problem formulation.

3.1 Notations

We first summarize the notations used in this paper. A network is defined as $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, where $\mathbf{V} = \{v_1, v_2, ..., v_N\}$ denotes a set of nodes and N is the number of the nodes. $\mathbf{E}$ is the set of edges between the nodes, and $M = |\mathbf{E}|$ is the number of the edges. In this paper, we mainly consider undirected networks. Let $\mathbf{Nbrs}_i = \{v_j | (v_i, v_j) \in \mathbf{E}\}$ denote the set of neighbors of node v_i. Let $\mathbf{P} \in \mathbb{R}^{N \times N}$ be the transition matrix, where $\mathbf{P}(i,:)$ and $\mathbf{P}(:,j)$ denote its i^{th} row and j^{th} column respectively and $\mathbf{P}(i,j)$ is the element of the i^{th} row and j^{th} column. If there is an edge from v_i to v_j and the degree of node v_i is d_i, then we set $\mathbf{P}(i,j)$ to $\frac{1}{d_i}$, otherwise we mark $\mathbf{P}(i,j)$ with zero. We define $\mathbf{h}_i = \mathcal{N}(\mu_i, \Sigma_i)$ as a lower-dimensional Gaussian distribution embedding for node v_i, where $\mu_i \in \mathbb{R}^L$, $\Sigma_i \in \mathbb{R}^{L \times L}$. L is the embedding dimension, which satisfies $L \ll N$. In this paper, we focus on diagonal covariance matrices.

3.2 Problem Definition

In this paper, we focus on the problem of network embedding with first-order and second-order proximity preserved.

Definition 3.1. (First-Order Proximity) The first-order proximity describes the pairwise proximity between nodes. For any pair of nodes, if $\mathbf{P}(i,j) > 0$, there exists positive first-order proximity between v_i and v_j. Otherwise, the first-order proximity between v_i and v_j is 0.

The first-order proximity implies that two nodes in real-world networks are similar if they are linked by an observed edge. For example, if two users build a relationship between them on the social network, they may have a common interest. However, real-world networks are usually so sparse that we can only observe a very limited number of links. Only capturing the first-order proximity is not sufficient,thus we introduce the second-order proximity to capture the global network structure.

Definition 3.2. (Second-Order Proximity) The second-order proximity between a pair of nodes denotes the similarity between their neighborhood network structures. Then the second-order proximity between v_i and v_j is determined by the similarity between $\mathbf{Nbrs}_i$ and $\mathbf{Nbrs}_j$. If none of nodes is linked with both v_i and v_j, the second-order proximity between v_i and v_j is 0.

Intuitively, the second-order proximity assumes that if two nodes share common neighbors, they tend to be similar. The second-order proximity has been demonstrated to be a good metric to define the similarity of a pair of nodes, even if there is no edge between them [31]. Moreover, the second-order proximity has been proved to be able to alleviate the sparsity problem of the first-order proximity and better preserve the global structure of the network [39].

With the first- and second-order proximity, then we define our network embedding problem as follows:

Definition 3.3. (Gaussian-Based Network Embedding) Given a network $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, we aim to represent each node v_i as a lower-dimensional Gaussian distribution $\mathbf{h}_i = \mathcal{N}(\mu_i, \Sigma_i)$, where μ_i captures the position of the nodes in the embedding space and Σ_i investigates the uncertainty of the nodes. Meanwhile, the latent representations aim to preserve the first-order proximity and the second-order proximity between the nodes to preserve the network structure.

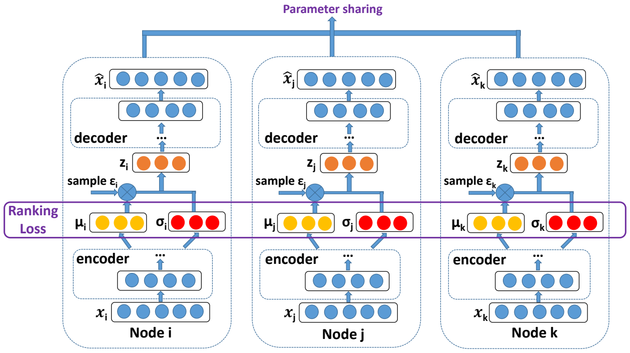


Figure 1: The framework of DVNE.

4 DEEP VARIATIONAL NETWORK EMBEDDING

4.1 Framework

In this paper, we propose a novel model to perform network embedding, namely DVNE, whose framework is shown in Figure 1. Basically, we propose a deep architecture, which is composed of multiple nonlinear mapping functions to map the input data to the Wasserstein space to preserve the uncertainties of the nodes and capture the network structure. Specifically, we first use a ranking based loss function on the Wasserstein embedding space, aiming to make nodes with edges similar and without edges dissimilar. In this way, the first-order proximity is preserved. Furthermore, we use a deep variational model to preserve the second-order proximity, by reconstructing the neighborhood structure of each node. Meanwhile, the whole deep variational model implies the the mathematical relevance of mean vectors and variance terms explicitly by the sampling process. In this way, the mean vectors find an approximate position of the node and the variance term capture the uncertainty. In the following sections, we will introduce how to realize the deep model in detail.

4.2 Similarity Measure

To support network applications, we need to define a suitable similarity measure between the latent representations of two nodes. Since we use distributions to represent our latent representations to incorporate uncertainty, the similarity measure should be able to measure the similarity between the distributions. Furthermore, as transitivity is a important property of the network, the similarity measure should simultaneously preserve the transitivity between nodes. Through extensive studies, we find that the Wasserstein distance is able to measure the similarity between two distributions while simultaneously satisfies the triangle inequality [9], which guarantees its ability to preserve the transitivity of similarity between nodes.

The p^{th} Wasserstein distance between two probability measures μ and ν is defined as:

$$W_p(\mu, \nu)^p = \inf \mathbb{E}\left[d(X, Y)^p\right], \tag{1}$$

where $\mathbb{E}[Z]$ denotes the expected value of a random variable Z and the infimum is taken over all joint distributions of the random variables X and Y with marginals μ and ν respectively. Moreover, when $p \geq 1$, the p^{th} Wasserstein distance preserves all properties of a metric [1], including both the symmetry and the triangle inequality [6]. In this way, Wasserstein distance is suitable to be a similarity measure between the latent representation of nodes, especially for an undirected network.

But the calculation of the general-formed Wasserstein distance is limited by a heavy computational cost, which poses a great challenge to network applications. To reduce the computational cost, in our case since we use Gaussian distributions for the latent representation of nodes, the 2^{th} Wasserstein distance (abbreviated as W_2) has the closed form solution to speed-up the calculation process. The W_2 distance has also been widely used in in computer vision [4, 11], computer graphics [5, 15] or machine learning [12, 14].

More specifically, we have the following formula to calculate W_2 distance between two Gaussian distributions [20]:

$$\begin{aligned} dist &= W_2(\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2)) \\ dist^2 &= \|\mu_1 - \mu_2\|_2^2 + \mathrm{Tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2})^{1/2}) \end{aligned} \tag{2}$$

In this paper we focus on diagonal covariance matrices[1], thus $\Sigma_1\Sigma_2 = \Sigma_2\Sigma_1$. Then the formula (2) can be simplified as:

$$W_2(\mathcal{N}(\mu_1, \Sigma_1); \mathcal{N}(m_2, \Sigma_2))^2 = \|\mu_1 - \mu_2\|_2^2 + \|\Sigma_1^{1/2} - \Sigma_2^{1/2}\|_F^2. \tag{3}$$

According to the above equation, the time complexity of calculating W_2 distance between the latent representation of two nodes is linear with the embedding dimension L. Therefore, we choose W_2 distance as the similarity measure, and the computational costs no longer constitute limitations.

4.3 Loss Functions

Our overall loss functions for DVNE consists of two parts, the ranking-based loss to preserve the first-order proximity and the reconstruction loss to preserve second-order proximity.

First, we consider how to preserve the first-order proximity. Intuitively, we want all nodes which are linked with v_i to be closer to v_i w.r.t. their embedding, compared to the nodes that have no edge with v_i. More specifically, we propose the following pairwise constraints to preserve the first-order proximity:

$$W_2(\mathbf{h}_i, \mathbf{h}_j) < W_2(\mathbf{h}_i, \mathbf{h}_k), \forall v_i \in \mathbf{V}, \forall v_j \in \mathbf{Nbrs}_i, \forall v_k \notin \mathbf{Nbrs}_i. \tag{4}$$

where $\mathbf{h}_i$ is the latent representation of node v_i, $\mathbf{Nbrs}_i$ is the set of neighbors of node v_i. The smaller the W_2 distance, the larger the similarities between nodes.

Then we use a energy based learning approach [29] to incorporate all of the pairwise constraints defined in the above equation. Mathematically, denoting $E_{ij} = W_2(\mathbf{h}_i, \mathbf{h}_j)$ as the energy between two nodes, we present the objective function as follows:

$$\mathcal{L}_1 = \sum_{(i,j,k) \in \mathbf{D}} ({E_{ij}}^2 + exp(-E_{ik})), \tag{5}$$

where $\mathbf{D}$ is the set of all valid triplets given in Eq. (4). The above objective function penalizes ranking errors by the energy of the pairs, which makes the energy of positive examples to be lower than that of negative examples. Equivalently, it will make the similarity between the positive examples larger than that of negative examples, thus helps preserve the first-order proximity.

For second-order proximity, we use the transition matrix $\mathbf{P}$ as our input features and propose a variant of Wasserstein Auto-Encoders (WAE) [41] as the model to preserve the neighborhood structure. WAE is a deep variational model, which can imply the mathematical relevance of mean vectors and variance terms by the sampling process. The objective of original WAE is composed of two terms, the reconstruction cost and the regularizer. The reconstruction cost aims to capture the information of the input. The regularizer encourages the encoded training distributions to match the prior distribution. As for our problem, the $\mathbf{P}(i, :)$ shows the neighborhood structure of node v_i, thus we use $\mathbf{P}(i, :)$ as the input feature to the WAE for node v_i and reconstruct it to preserve its neighborhood structure. For the regularization term, it is hard to define the prior distribution of each node in the network. Therefore, we focus only on the reconstruction cost to preserve the neighborhood structure.

Let P_X denote the data distribution, and P_G denote the encoded training distribution. The reconstruction cost can be represented as:

$$D_{WAE}(P_X, P_G) = \inf_{Q(Z|X) \in Q} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)} \big[c(X, G(Z))\big], \tag{6}$$

where Q is the encoders and G is the decoders, $X \sim P_X$ and $Z \sim Q(Z|X)$. It aims to minimize Wasserstein distance between the P_X and P_G.

According to [41], when using $c(x, y) = \|x - y\|_2^2$, the above loss function (6) minimizes the W_2 distance between P_X and P_G, thus P_G captures the information of the input data in the Wasserstein space.

Considering the sparsity of the transition matrix $\mathbf{P}$, we focus on non-zero elements in $\mathbf{P}$ to speed up our model. Thus, we present the loss function as follows to preserve the second-order proximity:

$$\mathcal{L}_2 = \inf_{Q(Z|X) \in Q} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)} \big[\|X \circ (X - G(Z))\|_2^2\big], \tag{7}$$

where $\circ$ means the element-wise multiplication.

In our model, we use the transition matrix $\mathbf{P}$ as the input feature X. The reconstruction process will make the nodes with similar neighborhoods have similar latent representations. Therefore, the second-order proximity between nodes is preserved.

To preserve first-order proximity and second-order proximity of networks simultaneously, we jointly minimize the loss function by combining Eq. (5) and Eq. (7):

$$\mathcal{L} = \mathcal{L}_1 + \alpha \mathcal{L}_2. \tag{8}$$

4.4 Optimization

For large graphs, optimizing objective function (5) is computationally expensive, which requires to calculate the all valid triplets in $\mathbf{D}$. Therefore, we sample triplets from $\mathbf{D}$ uniformly, which replace $\sum_{(i,j,k) \in \mathbf{D}}$ with $\mathbb{E}_{(i,j,k) \sim \mathbf{D}}$ in Eq. (5). In details, for each iteration, we sample M triplets from $\mathbf{D}$ to calculate the estimates of the gradient.

Considering objective function (7), we need sample Z from $Q(Z|X)$, which is a non-continuous operation and has no gradient. In this case, it is difficult for the deep models to optimize the loss

[1] When the covariance matrices is not diagonal, Wang proposed an fast iterative algorithm (called BADMM) to solve the Wasserstein distance [45]. It is not the focus of the paper and we will not discuss it.

function. To solve the problem, inspired by the Variational Auto-Encoders (VAE) [16], we can use the "reparameterization trick" to optimize the above objective equation. Mathematically, we first sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, then compute $Z = \mu(X) + \Sigma^{1/2}(X) * \epsilon$. Given a fixed X and ϵ, the objective function (7) is deterministic and continuous in the parameters of encoders Q and decoders G. In this way, the whole model can get the gradient when performing the back-propagation, and thus we can use stochastic gradient descent to optimize the model.

4.5 Implementation Details

For all the experiments in this paper we used an encoder and a decoder with a single hidden layer of size $S = 512$ respectively. More specifically, to obtain the embeddings for a node v_i, we have

$$\begin{aligned}
\mathbf{y}_i^{(1)} &= Relu(\mathbf{x}_i\mathbf{W}^{(1)} + \mathbf{b}^{(1)}), \mathbf{W}^{(1)} \in \mathbb{R}^{N\times S}, \mathbf{b}^{(1)} \in \mathbb{R}^S \\
\mu_i &= \mathbf{y}_i^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}, \mathbf{W}^{(2)} \in \mathbb{R}^{S\times L}, \mathbf{b}^{(2)} \in \mathbb{R}^L \\
\sigma_i &= Elu(\mathbf{y}_i^{(1)}\mathbf{W}^{(3)} + \mathbf{b}^{(3)}) + 1, \mathbf{W}^{(3)} \in \mathbb{R}^{S\times L}, \mathbf{b}^{(3)} \in \mathbb{R}^L \\
\mathbf{z}_i &= \mu_i + \sigma_i * \epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I}) \\
\mathbf{y}_i^{(2)} &= Relu(\mathbf{z}_i\mathbf{W}^{(4)} + \mathbf{b}^{(4)}), \mathbf{W}^{(4)} \in \mathbb{R}^{L\times S}, \mathbf{b}^{(4)} \in \mathbb{R}^S \\
\widehat{\mathbf{x}}_i &= Sigmoid(\mathbf{y}^{(2)}\mathbf{W}^{(5)} + \mathbf{b}^{(5)}), \mathbf{W}^{(5)} \in \mathbb{R}^{S\times N}, \mathbf{b}^{(5)} \in \mathbb{R}^N,
\end{aligned} \tag{9}$$

where x_i is $\mathbf{P}(i, :)$, Relu [34] and Elu [10] are the rectified linear unit and exponential linear unit. We use $elu() + 1$ to guarantee that σ_i is positive. Because the range of values in x_i is between [0, 1], we use the sigmoid function as the output function of the last hidden layer.

4.6 Complexity analysis

Algorithm 1 lists the procedures of our method. During the training procedure, the time complexity of calculating gradients and updating parameters is $O(T \times M \times (d_{ave}S + SL + L))$, where M is the number of the edges, d_{ave} is the average degree of all nodes, L is the dimension of embedding vectors, S is the size of hidden layer of the encoder and decoder, T is the number of iterations. Since we only reconstruct non-zero elements in x_i, the computational complexity of the first and last hidden layers is $O(d_{ave}S)$. The computational complexity of other hidden layers is $O(SL)$, and it takes $O(L)$ to calculate the W_2 distance between the distributions. In practice we found that a small number of iterations T ($T \leq 50$ for all shown experiments) is needed for convergence.

5 EXPERIMENT

In this section, we empirically evaluate the effectiveness of the our method.

5.1 Experiment Setting

We first introduce the experiment setting before presenting results of the experiments.

5.1.1 Baseline Methods. We use the following five methods as the baselines.

- DVNE_kl : In order to show the advantages of W_2 distance in undirected network. We replace the similarity measure in our method with the KL divergence.

Algorithm 1 Training algorithm of DVNE

```
Input: The network $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ with the transition matrix $\mathbf{P}$, the
    parameter $\alpha$
Output: Network embeddings $\{\mathbf{h}_i\}_{i=1}^N$ and updated parameters
    $\theta = \{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}\}_{i=1}^5$
Initial parameters $\theta$ by xavier initialization
while $\mathcal{L}$ do not converge do
    Sample $M$ triplets from $\mathbf{D}$ uniformly
    Split these triplets to a number of batches
    calculate partial derivative $\partial\mathcal{L}/\partial\theta$ with backpropagation
    algorithm to update $\theta$
end while
```

- DeepWalk [37]: This algorithm learns embedding by simulating several uniform random walks. It assumes that a pair of nodes are similar if they are close in the random walks.
- LINE [39]: This algorithm preserves the first-order and second-order proximity between nodes respectively, and directly concatenates the representations for the first-order and second-order proximity.
- SDNE [44]: This method learns a point-vector for each node with preserving the first and the second order proximities simultaneously using deep models.
- Graph2Gauss(G2G_oh) [2]: This method aims to learn the lower-dimensional Gaussian distribution embedding by ranking similarity based on the shortest path between nodes. As the datasets have no attribute information, we compare with the one-hot encoding version of Graph2Gauss as described in the paper.

5.1.2 Dataset. In order to comprehensively evaluate the effectiveness of our proposed method, we use four different real-world datasets, including citation networks and social networks. The detailed information is shown as follows:

- Cora : This is a research paper set constructed by McCallum et al. [33], which consists of 2708 scientific publications classified into one of seven classes.
- Facebook : It is a typical social network dataset without node labels constructed by J. McAuley et al. [30].
- BlogCatalog[38]: This is a network of social relationships of the bloggers listed on the BlogCatalog website. The labels represent the topic categories provided by the authors.
- Flickr [38]: It is a social network where node represents users and edges correspond to friendships between users. The labels represent the interest groups of the users.

All the networks are undirected, and the detailed statistics of the datasets are summarized in Table 1.

5.1.3 Parameter Settings. In all experiments, we set the embedding dimension $L = 128$ unless stated. For the equality, all the methods that learn the embedding as the distribution use the length of mean vector and variance terms to match L. Specifically, our method actually uses half of the dimensionality L as the length of mean vector in all experiments.

For DVNE and DVNE_kl, the hyper-parameters of α are tuned by using grid search on the validation set. We use xavier initialization

Table 1: Statistics of datasets. $|V|$ denotes the number of nodes , $|E|$ denotes the number of edges and $|C|$ denotes the number of classes.

	Cora	Facebook	BlogCatalog	Flickr
$\|V\|$	2,708	4,039	10,312	80,513
$\|E\|$	5,429	88,234	333,983	5,899,882
$\|C\|$	7	-	39	195

[21] for all weight matrices. The parameters are optimized using RMSProp [40] with a fixed learning rate of 0.001.

The parameters for baselines are tuned to be optimal. For DeepWalk, we set window size as 10, walk length as 40, walks per node as 10. For LINE, we set the number of negative samples as 5, and line search for the optimal value of the training samples on different datasets. For SDNE, we use the default parameter settings and the multi-layer deep structure in the author's implementation. For G2G_oh, we use the default parameter settings and the fixed learning rate in the implementation details of the paper.

5.2 Network Reconstruction

The most primal objective for network embedding is to reconstruct the given network, ans a good network embedding method should ensure that the learned embeddings can preserve the original network structure. Thus, we first provide a basic evaluation on different network embedding methods with respect to their capability of network reconstruction. More specifically, we use different network embedding methods to learn the embedding vectors on the different real-world networks. Then we rank pairs of nodes according to their trained similarities between the embedding of nodes, i.e. the W_2 distance for our method, the KL divergence for G2G_oh. The larger the similarities between pairs of nodes, the more likely they have the edges. Then we can use the top ranking pairs to reconstruct the edges of the original networks. For the evaluation metric, we use Area Under Curve (AUC) [18].

Table 2: AUC scores for Network Reconstruction.

	Cora	Facebook	BlogCatalog	Flickr
DVNE	**0.996**	**0.998**	**0.962**	**0.959**
DVNE_kl	0.940	0.958	0.937	0.925
DeepWalk	0.986	0.984	0.864	0.950
Line	0.952	0.934	0.891	0.939
SDNE	0.992	0.960	0.958	0.917
G2G_oh	0.921	0.942	0.924	0.901

The results are shown in Table 2. Our proposed method outperforms the baseline methods in all datasets. The results demonstrate that our proposed method can effectively preserve the original network structure and reconstruct the network. It lays the foundation for other real-world applications of network embedding.

5.3 Link Prediction

Link prediction, aiming to predict which pairs of nodes will form edges in the future, is a typical task of network embedding. In our experiments, we randomly hide 20% of the edges as the testing network and train the embeddings on the rest of the network. After the training, we can obtain the embedding for each node and then use the embeddings to predict the unobserved edges. The pairs of nodes are ranked in a similar way as network reconstruction and the top ranking pairs are evaluated on the testing network. Unlike the reconstruction task, this task predicts the unobserved edges in testing network instead of reconstructing the existing edges in training network. We still use AUC as the evaluation metric.

Table 3: AUC scores for Link Prediction.

	Cora	Facebook	BlogCatalog	Flickr
DVNE	**0.947**	**0.982**	**0.945**	**0.942**
DVNE_kl	0.919	0.930	0.917	0.908
DeepWalk	0.880	0.923	0.827	0.931
Line	0.854	0.882	0.802	0.919
SDNE	0.917	0.931	0.920	0.927
G2G_oh	0.901	0.925	0.903	0.906

From the results in Table 3, our proposed method still outperforms the baselines in all datasets. Especially on the facebook dataset, our method significantly improve AUC scores by 0.05 than the baselines. From the results, we have the following analysis:

Deepwalk can introduce high-order proximity by changing the parameter of window size, but it can not balance the weight of the first-order proximity and the high-order proximity. This means it can not handle well both reconstruction task and prediction task at the same time, which is evident from the experimental results. We also find that LINE does not achieve as good performance as other methods do in most cases. The reason may be twofold. Firstly, LINE adopts shallow structure, which is difficult to capture the highly non-linear structure [44] in the network. Moreover, LINE directly concatenates the embeddings for the first-order and second-order proximity, which is sub-optimal than jointly optimizing them in our method.

Although DVNE and SDNE both exploit the first-order and second-order proximity to preserve the network structure, DVNE achieves better performance. The reason is that our method learns a Gaussian distribution as an embedding for each node, allowing us to capture uncertainty in the network by the latent representations. Actually, adding a new edge between two nodes is a uncertain event, it is more natural to describe this event from the perspective of the distributions.

We also find that DVNE achieves a substantial gain over DVNE_kl on all the datasets. The reason is two fold. Firstly, the KL divergence is not suitable for undirected network because of the asymmetric property of the KL divergence. Secondly, the KL divergence does not necessarily guarantee the transitivity of similarities between the nodes, which makes KL-based methods worse link prediction results.

Compared with DVNE_kl and G2G_oh, which both use the KL divergence as the similarity measures, DVNE_kl outperforms G2G_oh. It is because that G2G_oh use the variance terms as the added dimensions while DVNE_kl relates the variance terms and the mean

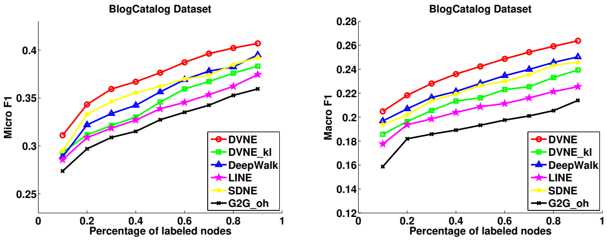


Figure 2: Micro-F1 and Macro-F1 on BlogCatalog.

vectors by the sampling process. Thus, DVNE is able to better capture the uncertainties of nodes and get a better link prediction result.

Overall, the results demonstrate that our proposed method works well for network inference tasks.

5.4 Multi-label Classification

Multi-label classification is another task commonly used to evaluate the effectiveness of the learned embeddings. We evaluate the multi-label classification performance for three datasets (Cora, Blogcatalog and Flickr) that have ground-truth labels. The representations for the nodes are generated from the network embedding methods and are used as features to classify each node into a set of labels. For all methods based on the distribution, we only use the mean vectors as the input features in this task. We adopt a linear SVC [23] as the classifiers for all methods. Then, following [37], we randomly sample a portion of the labeled nodes as the training data and the rest as the test. For BlogCatalog, we randomly sample 10% to 90% of the nodes as the training samples and use the left nodes to test the performance. For Cora and for Flickr, we randomly sample 1% to 10% of the nodes as the training samples and use the left nodes to test the performance on even more sparsely labeled networks. We use the Micro-F_1 and Macro-F_1 scores to evaluate the performance and report results averaged over 10 trials. The results are shown in Figure 2 and Figure 3 respectively.

In Figure 2 and Figure 3, the curve of our method is consistently above the curves of baseline methods. It demonstrates that our method can achieve a better classification performance than baselines even if the labelled data is limited. Such an advantage is meaningful for real-world applications, because the labelled data in real-world network is usually scarce. The variance terms of the representation can help us to deal with the noise information in the network, which makes the mean vectors to better capture the network structure. Therefore, the learned network embedding of our method can better generalize to the classification task than baselines.

In most cases, the performance of G2G_oh is the worst among all the compared network embedding methods. The reasons are two-fold. First, G2G_oh uses the variance terms as the added dimensions, causing part of the information of the proximity between nodes included in variance terms. In this way, the performance of G2G_oh greatly degrades. Our method, by using the deep variational model, makes the mean vectors and the variance terms capture different properties of the network, i.e. the mean vector captures the proximity and the variance term captures the uncertainty. In this way, our method can encode more proximity-based information into the mean vectors and thus perform much better than G2G_oh. Second, similar to the previous task, the KL divergence is not a suitable similarity measure to capture the transitivity for the undirected networks.

5.5 Embedding Uncertainty

Learning an embedding as a distribution rather than a point-vector allows us to capture uncertainty of the nodes. With our intuition, the nodes that have less links with other nodes, are harder to get a exact point-vector in the latent space. In other words, the lower the degree of a node, the less discriminative information it contains, thus making its embedding more uncertain. Then we conduct the following experiment to evaluate the intuition. For each node, we select its 10 dimensions with the largest variance and averaged the variance of the 10 dimensions as the variance value for the node. Then for each network dataset, we divide the total nodes into 10 parts based on their degrees. For each part of the nodes, we report the relationship between their degree and their averaged variance values. The Figure 4a shows the result on the all datasets. The horizontal axis represents the $\log_{10}()$ values of degree. Because the max degree of the node is no more than 200 in Cora, the line of Cora is different from the other datasets.

From Figure 4a, we find that the experimental results support our intuition. The nodes with higher degree contains rich information, thus making their variance smaller. Meanwhile, we can see that when the network is denser like Facebook and Flickr, the average variance of embeddings is smaller. This means that our learned embeddings of variance can reflect the density of the network.

Moreover, to demonstrate that the uncertainty in variance terms can help to deal with the noise edges in networks, we conduct an experiment to show the benefits of the uncertainty. First, following the setting in link prediction, we randomly hide 20% of the edges as the testing network and use the rest of network as the training network. Then we randomly choose some pairs of nodes as the noise edges and add them into the training network. We use different network embedding methods to learn the representations of nodes in the modified training network. Similar to link prediction task, we use the similarity between the learned node embeddings to predict the unobserved edges in testing network. We use the results of each method reported in link prediction as the benchmark to calculate the percentage of AUC decline. We vary the percentage of noise edges from 0.05 to 0.5, then show the percentage of AUC decline with respect to it in Figure 4b.

From the results shown in Figure 4b, we can see that the performance of our method is least affected by the noise edges. It demonstrates that our method can better deal with the noise edges in networks by capturing the uncertainties of the nodes. DeepWalk adopts random walk to generate network representations. Each node walks to other communities with a lower probability in the modified training network. Thus, DeepWalk can still preserve the original network structure and the result of DeepWalk is also good. DVNE_kl uses the KL divergence as the similarity measure, which can not well preserve the transitivity in the networks. The noise edges between nodes will further damage this property, leading to worse results. For G2G_oh, there is a weak connection between

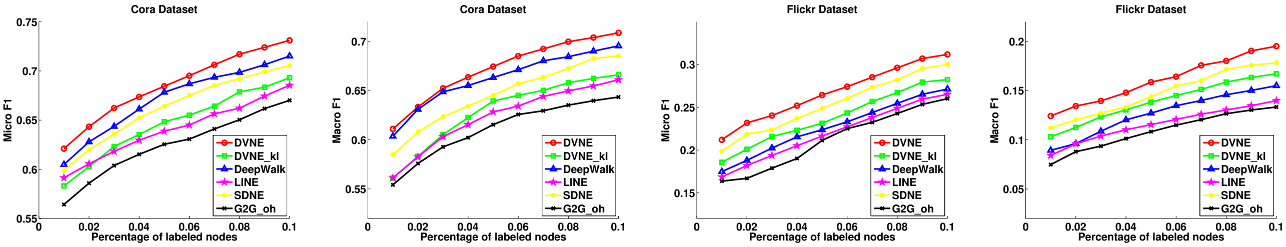
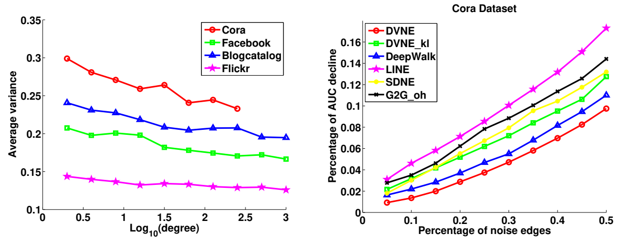


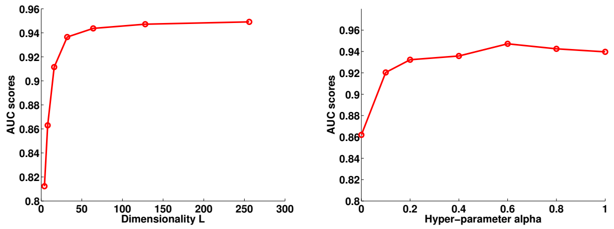
Figure 3: Micro-F1 and Macro-F1 on Cora and Flickr.



(a) The average variance wrt the degree of nodes.

(b) The performance wrt the noise edges.

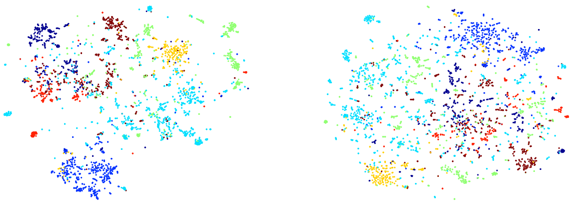
Figure 4: Results of embedding uncertainty.



(a) The AUC scores wrt the dimensionality L

(b) The AUC scores wrt the hyper-parameter α

Figure 6: Results of parameter sensitivity.



(a) DVNE

(b) DVNE_kl

Figure 5: Visualization of network embedding.

variance terms and mean vectors in the model, which means the variance terms can not well capture the uncertainties of the nodes. Through the sampling process proposed by our method, DVNE is more natural to learn the variance terms that contains the uncertainties of the nodes. Therefore, DVNE and DVNE_kl achieve better performance than G2G_oh. SDNE and LINE treat each edge equally, thus the similarities between nodes in latent space are easily be destroyed by the noise edges.

5.6 Visualization

Visualization is another important application for network embedding. Therefore, we visualize the learned embeddings of the Cora network. Following [39], we first learn a lower-dimensional $L = 128$ embedding for each node and then map those representations in 2-dimension space by t-SNE [32]. For nodes with different labels, we use different colors. Thus, a good visualization result is that the points of the same color are near from each other.

The visualization results are shown in Figure 5, we compare the DVNE with DVNE_kl. For DVNE_kl, in the center part the nodes of different classes are mixed with each other. Obviously, the visualization of DVNE looks better because points of the same color form segmented classes, and the boundaries of each class are clearer. It demonstrate the superiority of our method that using the W_2 distance as the similarity measure in the visualization task.

5.7 Parameter Sensitivity

In this section, we investigate the parameter sensitivity. More specifically, we evaluate how different numbers of the embedding dimensions and different values of hyper-parameter α can affect the results. We report AUC scores on the dataset of Cora.

First, we show how the dimension of the embedding vectors affects the performance in Figure 6a. We can see that initially the performance raises when the number of dimension increases. However, when the number of dimensions continuously increases, the performance tends to be stable. This is because most of the useful information is already encoded into the embeddings. Additional dimensions consume more computing resources, but have less effect on performance. Overall, it is important to determine the appropriate number of dimensions for the latent space. When the number of dimensions is not too small ($L \geq 32$), DVNE is not sensitive to this parameter.

Then, we fix the number of dimensions to 128. The Figure 6b shows how the value of α affects the performance . The parameter of α balances the weight of the first-order proximity and second-order proximity between nodes. When $\alpha = 0$, our method only preserves the first-order proximity between nodes and the performance is worse than that of other parameter settings. It demonstrates that both first-order and second-order proximity are essential for network embedding methods to capture the network structure. When $\alpha > 0$, we observe that DVNE is also not very sensitive to the choice of this hyper-parameter.

6 CONCLUSIONS

In this paper, we propose a method to learn the Gaussian embedding by the deep variational model, namely DVNE, which can model the uncertainties of nodes. It is the first unsupervised method that represents nodes in networks as Gaussian distributions in Wasserstein space. The method preserves first-order proximity and second-order proximity between nodes to capture the local and global network structure. Moreover, DVNE uses the 2-Wasserstein distance as the similarity measure to better preserve the transitivity in the network with the linear time complexity. The empirical study demonstrates the superiority of our proposed method. Our future direction is to find a good Gaussian prior for each node to better capture the network structure and model the uncertainties of nodes.

7 ACKNOWLEDGEMENTS

This work was supported in part by National Program on Key Basic Research Project (No. 2015CB352300), National Natural Science Foundation of China (No. 61772304, No. 61521002, No. 61531006, No. 61702296), National Natural Science Foundation of China Major Project (No.U1611461), the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, and the Young Elite Scientist Sponsorship Program by CAST. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. 2008. *Gradient flows: in metric spaces and in the space of probability measures.* Springer Science &amp; Business Media.

[2] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815* (2017).

[3] A. Bojchevski and S. Günnemann. 2017. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. *ArXiv e-prints* (July 2017). arXiv:stat.ML/1707.03815

[4] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. 2015. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision* 51, 1 (2015), 22–45.

[5] Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement interpolation using Lagrangian mass transport. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 158.

[6] Victor Bryant. 1985. *Metric spaces: iteration and application.* Cambridge University Press.

[7] Chen Chen and Hanghang Tong. 2015. Fast eigen-functions tracking on dynamic graphs. In *Proceedings of the 2015 SIAM International Conference on Data Mining.* SIAM, 559–567.

[8] Siheng Chen, Sufeng Niu, Leman Akoglu, Jelena Kovačević, and Christos Faloutsos. 2017. Fast, Warped Graph Embedding: Unifying Framework and One-Click Algorithm. *arXiv preprint arXiv:1702.05764* (2017).

[9] Philippe Clement and Wolfgang Desch. 2008. An elementary proof of the triangle inequality for the Wasserstein metric. *Proc. Amer. Math. Soc.* 136, 1 (2008), 333–339.

[10] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).

[11] Nicolas Courty, Rémi Flamary, and Mélanie Ducoffe. 2017. Learning Wasserstein Embeddings. *arXiv preprint arXiv:1710.07457* (2017).

[12] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. 2017. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence* 39, 9 (2017), 1853–1865.

[13] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2017. A Survey on Network Embedding. *arXiv preprint arXiv:1711.08752* (2017).

[14] Marco Cuturi and Arnaud Doucet. 2014. Fast computation of Wasserstein barycenters. In *International Conference on Machine Learning.* 685–693.

[15] Fernando De Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue noise through optimal transport. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 171.

[16] Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

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

[18] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[19] Bent Fuglede and Flemming Topsoe. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on.* IEEE, 31.

[20] Clark R Givens, Rae Michael Shortt, et al. 1984. A class of Wasserstein metrics for probability distributions. *The Michigan Mathematical Journal* 31, 2 (1984), 231–240.

[21] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics.* 249–256.

[22] 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.

[23] Steve R Gunn et al. 1998. Support vector machines for classification and regression. *ISIS technical report* 14, 1 (1998), 5–16.

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

[25] Paul W Holland and Samuel Leinhardt. 1972. Holland and Leinhardt reply: some evidence on the transitivity of positive interpersonal sentiment.

[26] Zhipeng Huang and Nikos Mamoulis. 2017. Heterogeneous Information Network Embedding for Meta Path based Proximity. *arXiv preprint arXiv:1701.05291* (2017).

[27] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[28] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[29] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. 2006. A tutorial on energy-based learning. *Predicting structured data* 1, 0 (2006).

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

[31] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology* 58, 7 (2007), 1019–1031.

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

[33] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[34] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10).* 807–814.

[35] Feiping Nie, Wei Zhu, and Xuelong Li. 2017. Unsupervised Large Graph Embedding.. In *AAAI.* 2422–2428.

[36] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proc. of ACM SIGKDD.* 1105–1114.

[37] 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.

[38] Zafarani Reza and Liu Huan. 2009. Social Computing Data Repository. (2009).

[39] 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.* ACM, 1067–1077.

[40] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[41] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. 2017. Wasserstein Auto-Encoders. *arXiv preprint arXiv:1711.01558* (2017).

[42] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. 2017. Structural Deep Embedding for Hyper-Networks. *arXiv preprint arXiv:1711.10146* (2017).

[43] Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623* (2014).

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

[45] Huahua Wang and Arindam Banerjee. 2014. Bregman alternating direction method of multipliers. In *Advances in Neural Information Processing Systems*. 2816–2824.

[46] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. (2017).

[47] Chengxi Zang, Peng Cui, Christos Faloutsos, and Wenwu Zhu. 2017. Long Short Memory Process: Modeling Growth Dynamics of Microscopic Social Connectivity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 565–574.