Structural Deep Network Embedding

Daixin Wang[1], Peng Cui[1], Wenwu Zhu[1]
[1]Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University. Beijing, China
dxwang0826@gmail.com,cuip@tsinghua.edu.cn,wwzhu@tsinghua.edu.cn

ABSTRACT

Network embedding is an important method to learn low-dimensional representations of vertexes in networks, aiming to capture and preserve the network structure. Almost all the existing network embedding methods adopt shallow models. However, since the underlying network structure is complex, shallow models cannot capture the highly non-linear network structure, resulting in sub-optimal network representations. Therefore, how to find a method that is able to effectively capture the highly non-linear network structure and preserve the global and local structure is an open yet important problem. To solve this problem, in this paper we propose a Structural Deep Network Embedding method, namely *SDNE*. More specifically, we first propose a semi-supervised deep model, which has multiple layers of non-linear functions, thereby being able to capture the highly non-linear network structure. Then we propose to exploit the first-order and second-order proximity jointly to preserve the network structure. The second-order proximity is used by the unsupervised component to capture the global network structure. While the first-order proximity is used as the supervised information in the supervised component to preserve the local network structure. By jointly optimizing them in the semi-supervised deep model, our method can preserve both the local and global network structure and is robust to sparse networks. Empirically, we conduct the experiments on five real-world networks, including a language network, a citation network and three social networks. The results show that compared to the baselines, our method can reconstruct the original network significantly better and achieves substantial gains in three applications, i.e. multi-label classification, link prediction and visualization.

Keywords

Network Embedding, Deep Learning, Network Analysis

1. INTRODUCTION

Nowadays, networks are ubiquitous and many real-world applications need to mine the information within these networks. For example, recommendation system in Twitter aims to mine the preferred tweets for users from the social network. Online advertisement targeting often needs to cluster the users into communities in the social network. Therefore, mining the information in the network is very important. One of the fundamental problems is how to learn useful network representations [5]. An effective way is to embed networks into a low-dimensional space, i.e. learn vector representations for each vertex, with the goal of reconstructing the network in the learned embedding space. As a result, mining information in networks, such as information retrieval [34], classification [15], and clustering [20], can be directly conducted in the low-dimensional space.

Learning network representations faces the following great challenges: (1) **High non-linearity**: As [19] stated, the underlying structure of the network is highly non-linear. Therefore, how to design a model to capture the *highly non-linear* structure is rather difficult. (2) **Structure-preserving**: To support applications analyzing networks, network embedding is required to preserve the network structure. However, the underlying structure of the network is very *complex* [24]. The similarity of vertexes is dependent on both the local and global network structure. Therefore, how to simultaneously preserve the local and global structure is a tough problem. (3) **Sparsity**: Many real-world networks are often so *sparse* that only utilizing the very limited observed links is not enough to reach a satisfactory performance [21].

In the past decades, many network embedding methods have been proposed, which adopted shallow models, such as IsoMAP [29], Laplacian Eigenmaps (LE) [1] and Line [26]. However, due to the limited representation ability of shallow models [2], it is difficult for them to capture the highly nonlinear network structure [30]. Although some methods adopt kernel techniques [32], as [36] stated, kernel methods are also shallow models and cannot capture the highly non-linear structure well.

In order to capture the **highly non-linear** structure well, in this paper we propose a new deep model to learn vertex representations for networks. This is motivated by the recent success of deep learning, which has been demonstrated to have a powerful representation ability to learn complex structures of the data [2] and has achieved substantial success in dealing with images [15], text [25] and audio [10] data. In particular, in our proposed model we design a multi-layer architecture which consists of multiple non-linear functions. The composition of multiple layers of non-linear functions can map the data into a highly non-linear latent space, thereby being able to capture the highly non-linear network structure.

In order to address the **structure-preserving** and **sparsity** problems in the deep model, we further propose to exploit the first-order and second-order proximity [26] jointly into the learning process. The first-order proximity is the local pairwise similarity only between the vertexes linked by edges, which characterizes the local network structure. However, due to the sparsity of the network,

KDD '16, August 13-17, 2016, San Francisco, CA, USA

DOI: http://dx.doi.org/10.1145/2939672.2939753

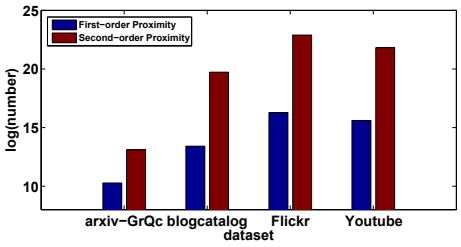


Figure 1: The number of pairs of vertexes which have first-order and second-order proximity in different datasets.

many legitimate links are missing. As a result, the first-order proximity is not sufficient to represent the network structure. Therefore, we further propose the second-order proximity, which indicates the similarity of the vertexes' neighborhood structures, to capture the global network structure. With the first-order and second-order proximity, we can well characterize the local and global network structure, respectively. To preserve both the local and global network structure in the deep model, we propose a semi-supervised architecture, in which the unsupervised component reconstructs the second-order proximity to preserve the global network structure while the supervised component exploits the first-order proximity as the supervised information to preserve the local network structure. As a result, the learned representations can well preserve both the local and global network structure. In addition, as shown in Figure 1, the number of pairs of vertexes which have second-order proximity is much huger than those have first-order proximity. Therefore, the import of second-order proximity is able to provide much more information in term of characterizing the network structure. As a result, our method is robust to sparse networks.

Empirically, we conduct the experiments on five real-world networked datasets and four real-world applications. The results show that compared with baselines, the representations generated by our method can reconstruct the original networks significantly better and achieve substantial gains on various tasks and various networks, including very sparse networks. It demonstrates that our representations learned in the highly non-linear space can preserve the network structure well and are robust to sparse networks.

In summary, the contributions of this paper are listed as follows:

- We propose a Structural Deep Network Embedding method, namely *SDNE*, to perform network embedding. The method is able to map the data to a highly non-linear latent space to preserve the network structure and is robust to sparse networks. To the best of our knowledge, we are among the first to use deep learning to learn network representations.

- We propose a new deep model with a semi-supervised architecture, which simultaneously optimizes the first-order and second-order proximity. As a result, the learned representations preserve the local and global network structure and are robust to sparse networks.

- The proposed method is extensively evaluated on five real datasets and four application scenarios. The results demonstrate the superior usefulness of the method in multi-label classification, reconstruction, link prediction and visualization. Specifically, our method can achieve more significant improvements (20%) over baselines when labelled data is scarce. In some cases we only need 60% less training samples but still achieve better performance.

2. RELATED WORK

2.1 Deep Neural Network

Representation learning has long been an important problem of machine learning and many works aim at learning representations for samples [3, 35]. Recent advances in deep neural networks have witnessed that they have powerful representations abilities [12] and can generate very useful representations for many types of data. For example, [15] proposed a seven-layer convolutional neural network to generate image representations for classification. [33] proposed a multimodal deep model to learn image-text unified representations to achieve cross-modality retrieval task.

However, to the best of our knowledge, there have been few deep learning works handling networks, especially learning network representations. In [9], Restricted Boltzmann Machines were adopted to do collaborative filtering. [30] adopted deep autoencoder to do graph clustering. [5] proposed a heterogeneous deep model to do heterogeneous data embedding. We differ from these works in two aspects. Firstly, the goals are different. Our work focuses on learning low-dimensional structure-preserved network representations which can be utilized among tasks. Secondly, we consider both the first-order and second-order proximity between vertexes to preserve the local and global network structure. But they only focus on one-order information.

2.2 Network Embedding

Our work solves the problem of network embedding, which aims to learn representations for networks. Some earlier works like Local Linear Embedding (LLE) [22], IsoMAP [29] first constructed the affinity graph based on the feature vectors and then solved the leading eigenvectors as the network representations. More recently, [26] designed two loss functions attempting to capture the local and global network structure respectively. Furthermore, [4] extended the work to utilize high-order information. Despite the success of these network embedding approaches, they all adopt shallow models. As we have explained earlier, it is difficult for shallow models to effectively capture the highly non-linear structure in the underlying network. In addition, although some of them attempt to use first-order and high-order proximity to preserve the local and global network structure, they learn the representations for them separately and simply concatenate the representations. Obviously, it is sub-optimal than simultaneously modeling them in a unified architecture to capture both the local and global network structure.

DeepWalk [21] combined random walk and skip-gram to learn network representations. Although empirically effective, it lacks a clear objective function to articulate how to preserve the network structure. It is prone to preserving only the second-order proximity. However, our method designs an explicit objective function, which aims at simultaneously preserving the local and global structure by preserving both the first-order and second-order proximity.

3. STRUCTURAL DEEP NETWORK EMBEDDING

In this section, we first define the problem. Then we introduce the proposed semi-supervised deep model of *SDNE*. At last we present some discussions and analysis on the model.

3.1 Problem Definition

We first give the definition of a Graph.

DEFINITION 1. *(Graph) A graph is denoted as $G = (V, E)$, where $V = \{v_1, ..., v_n\}$ represents n vertexes and $E = \{e_{i,j}\}_{i,j=1}^{n}$ represents the edges. Each edge $e_{i,j}$ is associated with a weight*

$s_{i,j} \geq 0$ [1]. *For v_i and v_j not linked by an edge, $s_{i,j} = 0$. Otherwise, for unweighted graph $s_{i,j} = 1$ and for weighted graph, $s_{i,j} > 0$.*

Network embedding aims to map the graph data into a low-dimensional latent space, where each vertex is represented as a low-dimensional vector and the network computing can be directly realized. As we have explained, both local and global structure are essential to be preserved. Then we first define the first-order proximity, which characterizes the local network structure.

DEFINITION 2. *(First-Order Proximity) The first-order proximity describes the pairwise proximity between vertexes. For any pair of vertexes, if $s_{i,j} > 0$, 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.*

Naturally, it is necessary for network embedding to preserve the first-order proximity because it implies that two vertexes in real-world networks are always similar if they are linked by an observed edge. For example, if a paper cites another paper, they should contain some common topic. However, real-world datasets are often so sparse that the observed links only account for a small portion. There exist many vertexes which are similar with each other but not linked by any edges. Therefore, only capturing the first-order proximity is not sufficient. We introduce the second-order proximity to capture the global network structure.

DEFINITION 3. *(Second-Order Proximity) The second-order proximity between a pair of vertexes describes the proximity of the pair's neighborhood structure. Let $\mathcal{N}_u = \{s_{u,1}, ..., s_{u,|V|}\}$ denote the first-order proximity between v_u and other vertexes. Then, second-order proximity is determined by the similarity of $\mathcal{N}_u$ and $\mathcal{N}_v$.*

Intuitively, the second-order proximity assumes that if two vertexes share many common neighbors, they tend to be similar. Such an assumption has been proved reasonable in many fields [6, 14]. For example, in linguistics words will be similar if they are always surrounded by similar contexts [6]. People will be friends if they have many common friends [14]. The second-order proximity has been demonstrated to be a good metric to define the similarity of a pair of vertexes, even if they are not linked by an edge [17], and thus can highly enrich the relationship of vertexes. Therefore, by introducing the second-order proximity, it is able to characterize the global network structure and alleviate the sparsity problem.

With the first-order and second-order proximity, we investigate the problem of how to integrate them simultaneously to preserve both the local and global structure when we perform network embedding. Such a problem is defined as follows:

DEFINITION 4. *(Network Embedding) Given a graph denoted as $G = (V, E)$, network embedding aims to learn a mapping function $f : v_i \longmapsto \mathbf{y_i} \in \mathbb{R}^d$, where $d \ll |V|$. The objective of the function is to make the similarity between $\mathbf{y_i}$ and $\mathbf{y_j}$ explicitly preserve the first-order and second-order proximity of v_i and v_j.*

3.2 The Model

3.2.1 Framework

In this paper, we propose a semi-supervised deep model to perform network embedding, whose framework is shown in Figure 2. In detail, to capture the highly non-linear network structure, we propose a deep architecture, which is composed of multiple non-linear mapping functions to map the input data to a highly non-linear latent space to capture the network structure. Furthermore, in order to address the structure-preserving and sparsity problems, we propose a semi-supervised model to exploit both the second-order and first-order proximity. For each vertex, we are able to obtain its neighborhood. Accordingly, we design the unsupervised component to preserve the second-order proximity, by reconstructing the neighborhood structure of each vertex. Meanwhile, for a small portion of pairs of nodes, we can obtain their pairwise similarities, i.e. the first-order proximities. Therefore, we design the supervised component to exploit the first-order proximity as the supervised information to refine the representations in the latent space. By jointly optimizing them in the proposed semi-supervised deep model, *SDNE* can preserve the highly-nonlinear local-global network structure well and is robust to sparse networks. In the following section, we will introduce how to realize the semi-supervised deep model in detail.

[1]For signed network, negative links exist. But in this paper we only consider non-negative links.

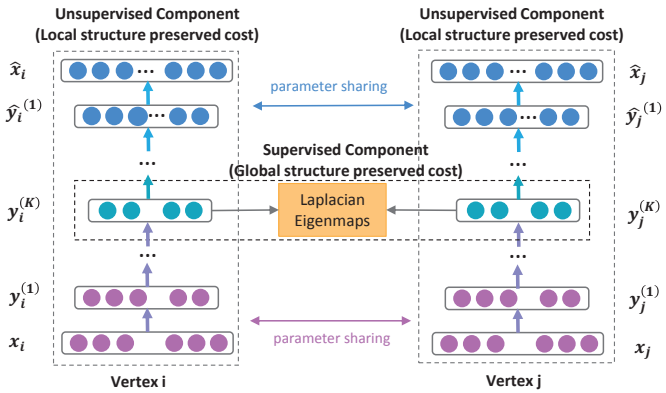


Figure 2: The framework of the semi-supervised deep model of *SDNE*

3.2.2 Loss Functions

Before introducing the loss functions, we define some of the terms and notations in Table 1 which will be used later. Note that $\hat{}$ above the parameters represents the parameters of the decoder.

Table 1: Terms and Notations

Symbol	Definition
n	number of vertexes
K	number of layers
$S = \{\mathbf{s}_1, ..., \mathbf{s}_n\}$	the adjacency matrix for the network
$X = \{\mathbf{x}_i\}_{i=1}^n, \hat{X} = \{\hat{\mathbf{x}}_i\}_{i=1}^n$	the input data and reconstructed data
$Y^{(k)} = \{\mathbf{y}_i^{(k)}\}_{i=1}^n$	the k-th layer hidden representations
$W^{(k)}, \hat{W}^{(k)}$	the k-th layer weight matrix
$\mathbf{b}^{(k)}, \hat{\mathbf{b}}^{(k)}$	the k-th layer biases
$\theta = \{W^{(k)}, \hat{W}^{(k)}, \mathbf{b}^{(k)}, \hat{\mathbf{b}}^{(k)}\}$	the overall parameters

Now we introduce the loss functions for the semi-supervised model. We first describe how the unsupervised component exploits the second-order proximity to preserve the global network structure.

The second-order proximity refers to how similar the neighborhood structure of a pair of vertexes is. Thus, to model the second-order proximity, it is required to model the neighborhood of each vertex. Given a network $G = (V, E)$, we can obtain its adjacency matrix S, which contains n instances $\mathbf{s}_1, ..., \mathbf{s}_n$. For each instance $\mathbf{s}_i = \{s_{i,j}\}_{j=1}^n$, $s_{i,j} > 0$ if and only if there exists a link between v_i and v_j. Therefore, $\mathbf{s}_i$ describes the neighborhood structure of the vertex v_i and S provides the information of the neighborhood structure of each vertex. With S, we extend the traditional deep autoencoder [23] to preserve the second-order proximity.

For the consideration of being self-contained, we briefly review the key idea of deep autoencoder. It is an unsupervised model

which is composed of two parts, i.e. the encoder and decoder. The encoder consists of multiple non-linear functions that map the input data to the representation space. The decoder also consists of multiple non-linear functions mapping the representations in representation space to reconstruction space. Then given the input $\mathbf{x}_i$, the hidden representations for each layer are shown as follows [2]:

$$\begin{aligned}\mathbf{y}_i^{(1)} &= \sigma(W^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}) \\ \mathbf{y}_i^{(k)} &= \sigma(W^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}), k = 2, ..., K\end{aligned} \tag{1}$$

After obtaining $\mathbf{y}_i^{(K)}$, we can obtain the output $\hat{\mathbf{x}}_i$ by reversing the calculation process of encoder. The goal of the autoencoder is to minimize the reconstruction error of the output and the input. The loss function is shown as follows:

$$\mathcal{L} = \sum_{i=1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 \tag{2}$$

As [23] proved, although minimizing the reconstruction loss does not explicitly preserve the similarity between samples, the reconstruction criterion can smoothly capture the data manifolds and thus preserve the similarity between samples. Then considering our case that if we use the adjacency matrix S as the input to the autoencoder, i.e. $\mathbf{x}_i = \mathbf{s}_i$, since each instance $\mathbf{s}_i$ characterizes the neighborhood structure of the vertex v_i, the reconstruction process will make the vertexes which have similar neighborhood structures have similar latent representations.

Nevertheless, such a reconstruction process cannot be directly applied to our problem because of some specific characteristics of networks. In the networks, we can observe some links but simultaneously many legitimate links are not observed, which means that the links between vertexes do indicate their similarity but no links do not necessarily indicate their dissimilarity. Moreover, due to the sparsity of networks, the number of non-zero elements in S is far less than that of zero elements. Then if we directly use S as the input to the traditional autoencoder, it is more prone to reconstruct the zero elements in S. However, this is not what we want. To address this problem, we impose more penalty to the reconstruction error of the non-zero elements than that of zero elements. The revised objective function is shown as follows:

$$\begin{aligned}\mathcal{L}_{2nd} &= \sum_{i=1}^{n} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{b_i}\|_2^2 \\ &= \|(\hat{X} - X) \odot B\|_F^2\end{aligned} \tag{3}$$

where $\odot$ means the Hadamard product, $\mathbf{b_i} = \{b_{i,j}\}_{j=1}^n$. If $s_{i,j} = 0$, $b_{i,j} = 1$, else $b_{i,j} = \beta > 1$. Now by using the revised deep autoencoder with the adjacency matrix S as input, the vertexes which have similar neighborhood structure will be mapped near in the representations space, guaranteed by the reconstruction criterion. In other words, the unsupervised component of our model can preserve the global network structure by reconstructing the second-order proximity between vertexes.

It is not only necessary to preserve the global network structure, but also essential to capture the local structure. We use the first-order proximity to denote the local network structure. The first-order proximity can be regarded as the supervised information to constrain the similarity of the latent representations of a pair of vertexes. Therefore, we design the supervised component to exploit the first-order proximity. The loss function for this goal is defined as follows [3] :

$$\begin{aligned}\mathcal{L}_{1st} &= \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)}\|_2^2 \\ &= \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2\end{aligned} \tag{4}$$

The objective function of Eq. 4 borrows the idea of Laplacian Eigenmaps [1], which incurs a penalty when similar vertexes are mapped far away in the embedding space. Some works about social networks [13] also use the similar idea. We differentiate them in the aspect that we incorporate the idea in the deep model to make the vertexes linked by an edge be mapped near in the embedding space. As a result, the model preserves the first-order proximity.

To preserve the first-order and second-order proximity simultaneously, we propose a semi-supervised model, which combines Eq. 4 and Eq. 3 and joint minimizes the following objective function:

$$\begin{aligned}\mathcal{L}_{mix} &= \mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \nu\mathcal{L}_{reg} \\ &= \|(\hat{X} - X) \odot B\|_F^2 + \alpha \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 + \nu\mathcal{L}_{reg}\end{aligned} \tag{5}$$

where $\mathcal{L}_{reg}$ is an $\mathcal{L}2$-norm regularizer term to prevent overfitting, which is defined as follows:

$$\mathcal{L}_{reg} = \frac{1}{2}\sum_{k=1}^{K}(\|W^{(k)}\|_F^2 + \|\hat{W}^{(k)}\|_F^2)$$

3.2.3 Optimization

To optimize the aforementioned model, the goal is to minimize $\mathcal{L}_{mix}$ as a function of θ. In detail, the key step is to calculate the partial derivative of $\partial\mathcal{L}_{mix}/\partial\hat{W}^{(k)}$ and $\partial\mathcal{L}_{mix}/\partial W^{(k)}$. The detailed mathematical form of the partial derivative is shown as follows:

$$\begin{aligned}\frac{\partial\mathcal{L}_{mix}}{\partial\hat{W}^{(k)}} &= \frac{\partial\mathcal{L}_{2nd}}{\partial\hat{W}^{(k)}} + \nu\frac{\partial\mathcal{L}_{reg}}{\partial\hat{W}^{(k)}} \\ \frac{\partial\mathcal{L}_{mix}}{\partial W^{(k)}} &= \frac{\partial\mathcal{L}_{2nd}}{\partial W^{(k)}} + \alpha\frac{\partial\mathcal{L}_{1st}}{\partial W^{(k)}} + \nu\frac{\partial\mathcal{L}_{reg}}{\partial W^{(k)}}, k = 1, ..., K\end{aligned} \tag{6}$$

We first look at $\partial\mathcal{L}_{2nd}/\partial\hat{W}^{(K)}$. It can be rephrased as follows:

$$\frac{\partial\mathcal{L}_{2nd}}{\partial\hat{W}^{(K)}} = \frac{\partial\mathcal{L}_{2nd}}{\partial\hat{X}} \cdot \frac{\partial\hat{X}}{\partial\hat{W}^{(K)}} \tag{7}$$

For the first term, according to Eq. 3 we have:

$$\frac{\partial\mathcal{L}_{2nd}}{\partial\hat{X}} = 2(\hat{X} - X) \odot B \tag{8}$$

The calculation of the second term $\partial\hat{X}/\partial\hat{W}$ is easy since $\hat{X} = \sigma(\hat{Y}^{(K-1)}\hat{W}^{(K)} + \hat{b}^{(K)})$. Then $\partial\mathcal{L}_{2nd}/\partial\hat{W}^{(K)}$ is accessible. Based on back-propagation, we can iteratively obtain $\partial\mathcal{L}_{2nd}/\partial\hat{W}^{(k)}, k = 1, ...K-1$ and $\partial\mathcal{L}_{2nd}/\partial W^{(k)}, k = 1, ...K$. Now the calculation of the partial derivative of $\mathcal{L}_{2nd}$ is finished.

Then we continue to calculate the partial derivative of $\partial\mathcal{L}_{1st}/\partial W^{(k)}$. The loss function of $\mathcal{L}_{1st}$ can be rephrased as follows:

$$\mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 = 2tr(Y^T L Y) \tag{9}$$

[2] In this work, we use the sigmoid function $\sigma(x) = \frac{1}{1+exp(-x)}$ as the non-linear activation function

[3] For simplicity of notations, we denote network representations $Y^{(K)} = \{\mathbf{y}_i^{(K)}\}_{i=1}^n$ as $Y = \{\mathbf{y}_i\}_{i=1}^n$.

where $L = D - S$, $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix, $D_{i,i} = \sum_j s_{i,j}$.

Then we first focus on the calculation of $\partial \mathcal{L}_{1st}/\partial W^{(K)}$:

$$\frac{\partial \mathcal{L}_{1st}}{\partial W^{(K)}} = \frac{\partial \mathcal{L}_{1st}}{\partial Y} \cdot \frac{\partial Y}{\partial W^{(K)}} \tag{10}$$

Since $Y = \sigma(Y^{(K-1)}W^{(K)} + b^{(K)})$, the calculation of the second term $\partial Y/\partial W^{(K)}$ is easy. For the first term of $\partial \mathcal{L}_{1st}/\partial Y$, we have:

$$\frac{\partial \mathcal{L}_{1st}}{\partial Y} = 2(L + L^T) \cdot Y \tag{11}$$

Similarly, by using back-propagation we can finish the calculation of partial derivative of $\mathcal{L}_{1st}$.

Now we have obtained the partial derivatives of the parameters. With an initialization of the parameters, the proposed deep model can be optimized by using stochastic gradient descent. Note that due to the high nonlinearity of the model, it suffers from many local optimal in the parameter space. Therefore, in order to find a good region of parameter space, we use Deep Belief Network to pretrain the parameters at first [11], which has been demonstrated as an essential initialization of parameters for deep learning in literature [7]. The full algorithm is presented in Alg. 1.

Algorithm 1 Training Algorithm for the semi-supervised deep model of *SDNE*

Input: the network $G = (V, E)$ with adjacency matrix S, the parameters α and ν

Output: Network representations Y and updated Parameters: θ

1: Pretrain the model through deep belief network to obtain the initialized parameters $\theta = \{\theta^{(1)}, ..., \theta^{(K)}\}$
2: $X = S$
3: **repeat**
4: Based on X and θ, apply Eq. 1 to obtain $\hat{X}$ and $Y = Y^K$.
5: $\mathcal{L}_{mix}(X; \theta) = \|(\hat{X} - X) \odot B\|_F^2 + 2\alpha tr(Y^T L Y) + \nu \mathcal{L}_{reg}$.
6: Based on Eq. 6, use $\partial L_{mix}/\partial \theta$ to back-propagate through the entire network to get updated parameters θ.
7: **until** converge
8: Obtain the network representations $Y = Y^{(K)}$

3.3 Analysis and Discussions

In this section, we present some analysis and discussions of the proposed semi-supervised deep model of *SDNE*.

New vertexes: A practical issue for network embedding is how to learn representations for newly arrived vertexes. For a new vertex v_k, if its connections to the existing vertexes is known, we can obtain its adjacency vector $\mathbf{x} = \{s_{1,k}, ..., s_{n,k}\}$, where $s_{i,k}$ indicates the similarity between the existing v_i and the new vertex v_k. Then we can simply feed the $\mathbf{x}$ into our deep model and use the trained parameters θ to get the representations for v_k. The complexity for such a process is $O(1)$. If there exist no connections between v_i and the existing vertexes in the network, our method and state-of-the-art network embedding methods cannot handle. To handle such case, we can resort to other side information, such as the content features of the new vertexes, which we leave as the future work.

Training Complexity: It is not difficult to see that the training complexity of our model is $O(ncdI)$, where n is the number of vertexes, d is the maximum dimension of the hidden layer, c is the average degree of the network and I is the number of iterations. Parameter d is usually related to the dimension of embedding vectors but not related to the number of vertexes. I is also independent with n. For c, it usually can be regarded as a constant in real-world applications. For example, in the social network, the maximum number of friends for a people is always bounded [30]. In top-k similarity graph, $c = k$. Therefore, cdI is independent with n and thus the overall training complexity is linear to the number of vertexes in the network.

4. EXPERIMENTS

In this section, we evaluate our proposed method on several real-world datasets and applications. The experimental results demonstrate significant improvements over baselines.

4.1 Datasets

In order to comprehensively evaluate the effectiveness of the representations, we use five networked datasets, including three social networks, one citation network and one language network, for three real-world applications, i.e. multi-label classification, link prediction and visualization. Considering the characteristics of these datasets, for each application we use one or more datasets to evaluate the performances. The detailed descriptions are listed as follows.

- BLOGCATALOG [27], FLICKR [27] and YOUTUBE [28]: They are social network of online users. Each user is labelled by at least one category. There are overall 39 different categories for BLOGCATALOG, 195 categories for FLICKR and 47 categories for categories. These categories can be used as the ground-truth of each vertex. Therefore, they can be evaluated on the multi-label classification task.

- ARXIV GR-QC [16]: It is a paper collaboration network which covers papers in the field of General Relativity and Quantum Cosmology from arXiv. In this network, the vertex represents an author and the edge indicates that the authors have coauthored a scientific paper in arXiv. The dataset is used for the link-prediction task since we have no category information of each vertex.

- 20-NEWSGROUP[4]: This dataset is a collection of approximate 20000 newsgroup documents and each document is labelled by one of the 20 different groups. We use the tf-idf vectors of each word to represent the document and the cosine similarity as the similarity between two documents. We can construct the network according to such similarities. We select the documents labelled as *comp.graphics*, *rec.sport.baseball* and *talk.politics.gums* to perform the visualization task.

To summarize, we conduct experiments on both weighted and unweighted, sparse and dense, small and large networks. Therefore, the datasets can comprehensively reflect the characteristics of the network embedding methods. The detailed statistics of the datasets can be summarized in Table 2.

Table 2: Statistics of the dataset

Dataset	#(V)	#(E)
BLOGCATALOG	10312	667966
FLICKR	80513	11799764
YOUTUBE	1138499	5980886
ARXIV GR-QC	5242	28980
20-NEWSGROUP	1720	Full-connected

[4]http://qwone.com/ jason/20Newsgroups/

4.2 Baseline Algorithms

We use the following five methods as the baselines. The first four are network embedding methods. *Common Neighbor* directly predicts the links over the networks, which has been demonstrated to be an effective method to perform link prediction [17].

- *DeepWalk* [21]: It adopts random walk and skip-gram model to generate network representations.
- *LINE* [26]: It defines loss functions to preserve the first-order or second-order proximity separately. After optimizing the loss functions, it concatenates these representations.
- *GraRep* [4]: It extends to high-order proximity and uses the SVD to train the model. It also directly concatenates the representations of first-order and high-order.
- *Laplacian Eigenmaps* (*LE*) [1]: It generates network representations by factorizing the Laplacian matrix of the adjacency matrix. It only exploits the first-order proximity to preserve the network structure.
- *Common Neighbor* [17]: It only uses the number of common neighbors to measure the similarity between vertexes. It is used as the baseline only in the task of link prediction.

4.3 Evaluation Metrics

In our experiment, we perform the task of reconstruction, link prediction, multi-label classification and visualization. For reconstruction and link prediction, we use *precision@k* and *Mean Average Precision (MAP)* to evaluate the performance. Their definitions are listed as follows:

- *precision@k* is a metric which gives equal weight to the returned instance. It is defined as follows:

$$precision@k(i) = \frac{\mid \{j \mid i, j \in V, index(j) \leq k, \Delta_i(j) = 1\} \mid}{k}$$

  where V is the vertex set, $index(j)$ is the ranked index of the j-th vertex and $\Delta_i(j) = 1$ indicates that v_i and v_j have a link.

- *Mean Average Precision (MAP)* is a metric with good discrimination and stability. Compared with *precision@k*, it is more concerned with the performance of the returned items ranked ahead. It is calculated as follows:

$$\begin{aligned} AP(i) &= \frac{\sum_j precision@j(i) \cdot \Delta_i(j)}{\mid \{\Delta_i(j) = 1\} \mid} \\ MAP &= \frac{\sum_{i \in Q} AP(i)}{\mid Q \mid}, \end{aligned}$$

  where Q is the query set.

For the multi-label classification task, we adopt *micro-*$F1$ and *macro-*$F1$ as many other works do [27]. In detail, for a label A, we denote TP(A), FP(A) and FN(A) as the number of true positives, false positives and false negatives in the instances which are predicted as A, respectively. Suppose $\mathcal{C}$ is the overall label set. *Micro-*$F1$ and *Macro-*$F1$ are defined as follows:

- *Macro-*$F1$ is a metric which gives equal weight to each class. It is defined as follows:

$$Macro - F1 = \frac{\sum_{A \in \mathcal{C}} F1(A)}{\mid \mathcal{C} \mid}$$

  where $F1(A)$ is the $F1$-measure for the label A.

- *Micro-*$F1$ is a metric which gives equal weight to each instance. It is defined as follows:

$$Pr = \frac{\sum_{A \in \mathcal{C}} TP(A)}{\sum_{A \in \mathcal{C}} (TP(A)+FP(A))}, R = \frac{\sum_{A \in \mathcal{C}} TP(A)}{\sum_{A \in \mathcal{C}} (TP(A)+FN(A))}$$
$$Micro - F1 = \frac{2 * Pr * R}{Pr + R}$$

4.4 Parameter Settings

We propose a multi-layer deep structure in this paper and the number of layers varies with different datasets. The dimension of each layer is listed in Table 3. The neural networks have three layers for BLOGCATALOG, ARXIV GR-QC and 20-NEWSGROUP and four layers for FLICKR and YOUTUBE. If we use deeper model, the performance almost remains unchanged or even becomes worse.

Table 3: Neural Network Structures

Dataset	#nodes in each layer
BLOGCATALOG	10300-1000-100
FLICKR	80513-5000-1000-100
YOUTUBE	22693-5000-1000-100
ARXIV GR-QC	5242-500-100
20-NEWSGROUP	1720-200-100

For our method, the hyper-parameters of α, β and ν are tuned by using grid search on the validation set. The parameters for baselines are tuned to be optimal. For *LINE*, the *mini-batch size* of the stochastic gradient descent is set to 1. The *learning rate* of the starting value is 0.025. The *number of negative samples* is set as 5 and the *total number of samples* is 10 billion. In addition, according to [26], *LINE* yields better results when concatenating both 1-step and 2-step representations to form the final embedding vectors and do $\mathcal{L}2$ normalization to the final embedding vectors. We follow their way to get the results of the *LINE*. For *DeepWalk*, we set *window size* as 10, *walk length* as 40, *walks per vertex* as 40. For *GraRep*, we set *maximum matrix transition step* as 5.

4.5 Experiment Results

In this section, we first evaluate the reconstruction performance. Then we report the results of the generalization of the network representations generated by different embedding methods on three classic data mining and machine learning applications, i.e. multi-label classification, link prediction and visualization.

4.5.1 Network Reconstruction

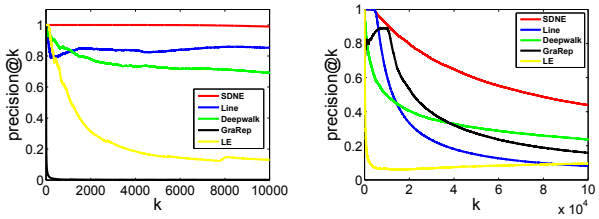
Before proceeding to evaluate the generalization of the proposed method on real-world applications, we first provide a basic evaluation on different network embedding methods with respect to their capability of network reconstruction. The reason for this experiment is that a good network embedding method should ensure that the the learned embeddings can preserve the original network structure. We use a language network ARXIV GR-QC and a social network BLOGCATALOG as representatives. Given a network, we use different network embedding methods to learn the network representations and then predict the links of the original networks. As the existing links in the original network are known and can serve as the ground-truth, we can evaluate the reconstruction performance, i.e. the training set error, of different methods. The *precision@k* and *MAP* are used as the evaluation metrics. The result on the *precision@k* is presented in Figure 3. The result on *MAP* is shown in Table 4.

From the results, we have the following observations and analysis:

Table 4: MAP on ARXIV-GRQC and BLOGCATALOG on reconstruction task

Method	ARXIV-GRQC					BLOGCATALOG				
	SDNE	*GraRep*	*LINE*	*DeepWalk*	*LE*	*SDNE*	*GraRep*	*LINE*	*DeepWalk*	*LE*
MAP	**0.836****	0.05	0.69	0.58	0.23	**0.63****	0.42	0.58	0.28	0.12

Significantly outperforms GraRep at the: ** 0.01 level.



(a) ARXIV GR-QC (b) BLOGCATALOG

Figure 3: precision@k on (a) ARXIV GR-QC and (b) BLOGCATALOG. The results show that our method achieves better reconstruction performance than that of baselines.

- Table 4 shows that our method achieves significant improvements in *MAP* over the baselines in both datasets. Figure 3 shows that the *precision@k* of our method is consistently the highest when k increases. It demonstrates that our method is able to preserve the network structure well.
- Specifically, for the network of ARXIV GR-QC, the *precision@k* of our method can reach 100% and maintain around 100% until the k increases to 10000. This indicates that our method can almost perfectly reconstruct the original network in this dataset, especially considering that the total number of links in this dataset is 28980.
- Although *SDNE* and *LINE* both exploit the first-order and second-order proximity to preserve the network structure, *SDNE* achieves better performance. The reasons may be twofold. Firstly, *LINE* adopts shallow structure, which is difficult to capture the highly non-linear structure in the underlying network. Secondly, *LINE* directly concatenates the representations for the first-order and second-order proximity, which is sub-optimal than jointly optimizing them in *SDNE*.
- The result that both *SDNE* and *LINE* perform better than *LE*, which only exploits the first-order proximity to preserve the network structure, demonstrates that the import of second-order proximity can help preserve the network structure better.

4.5.2 Multi-label Classification

Classification is a so important task among many applications that the related algorithm and theories have been investigated by many works [18]. Therefore, we evaluate the effectiveness of different network representations through a multilabel classification task in this experiment. The representations for the vertexes are generated from the network embedding methods and are used as features to classify each vertex into a set of labels. Specifically, we adopt the LIBLINEAR package [8] to train the classifiers. When training the classifier, 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 vertexes as the training samples and use the left vertexes to test the performance. For FLICKR and for YOUTUBE, we randomly sample 1% to 10% of the vertexes as the training samples and use the left vertexes to test the performance. In addition, we remove the vertexes which are not labelled by any categories in YOUTUBE. We repeat such a process 5 times and report the averaged *Micro-F1* and *Macro-F1*. The results are shown in Figure 4 and Figure 5, respectively.

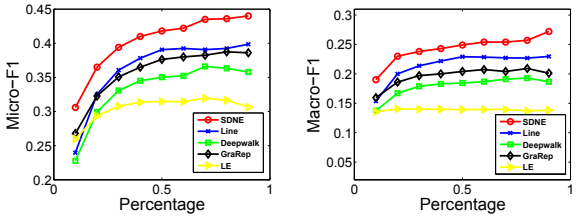


Figure 4: Micro-F1 and Macro-F1 on BLOGCATALOG. The results show that our method achieves better classification performance than that of baselines.

From the results, we have the following observations and analysis[5]:

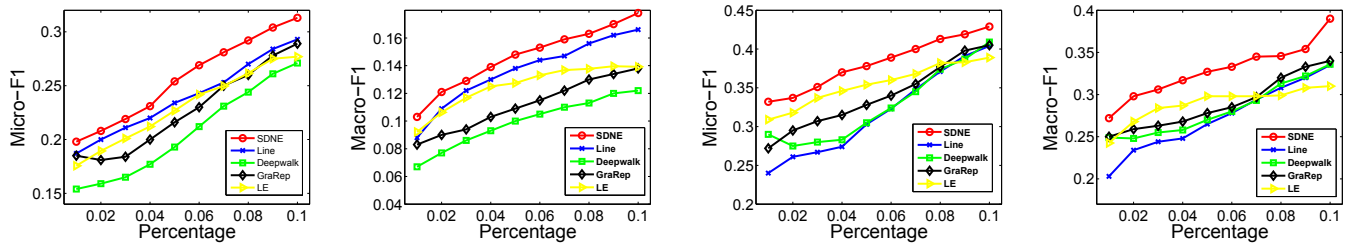
- In Figure 4 and Figure 5, the curve of our method is consistently above the curves of baselines. It demonstrates that the learned network representations of our method can better generalize to the classification task than baselines.
- In Figure 4 (BLOGCATALOG), the improvement margin of our method over the baselines is more obvious when the training percentage decreases from 60% to 10%. It demonstrates that our method can achieve a more significant improvement than baselines when the labelled data is limited. Such an advantage is especially important for real-world applications because the labelled data is usually scarce.
- In most cases, the performance of *DeepWalk* is the worst among the network embedding methods. The reasons are twofold. First of all, *DeepWalk* does not have an explicit objective function to capture the network structure. Secondly, *DeepWalk* uses a random walk to enrich the neighbors of vertexes, which introduces a lot of noises due to the randomness, especially for vertexes which have high degrees.

4.5.3 Link Prediction

In this section, we concentrate on the link prediction task and conduct two experiments. The first evaluates the overall performance and the second evaluates that how different sparsity of the networks affects the performance of different methods.

We use the dataset ARXIV GR-QC in this section. To conduct the link prediction task in a network, we randomly hide a portion of the existing links and use the left network to train the network embedding model. After the training, we can obtain the representations for each vertex and then use the obtained representations

[5]Some results such as the observation that our method outperforms *LINE* have been listed and explained in Section 4.5.1. Therefore, we only list some particular observations of this experiment.



(a) Micro-F1 and Macro-F1 on FLICKR

(b) Micro-F1 and Macro-F1 on YOUTUBE

Figure 5: Micro-F1 and Macro-F1 on (a) FLICKR and (b) YOUTUBE. The results show that our method achieves the best classification performance among baselines.

Table 5: *precision@k* on ARXIV GR-QC for link prediction

Algorithm	$P@2$	$P@10$	$P@100$	$P@200$	$P@300$	$P@500$	$P@800$	$P@1000$	$P@10000$
SDNE	**1**	**1**	**1**	**1**	**1***	**0.99****	**0.97****	**0.91****	**0.257****
LINE	1	1	1	1	0.99	0.936	0.74	0.79	0.2196
DeepWalk	1	0.8	0.6	0.555	0.443	0.346	0.2988	0.293	0.1591
GraRep	1	0.2	0.04	0.035	0.033	0.038	0.035	0.035	0.019
Common Neighbor	1	1	1	0.96	0.9667	0.98	0.8775	0.798	0.192
LE	1	1	0.93	0.855	0.827	0.66	0.468	0.391	0.05

Significantly outperforms Line at the: ** 0.01 and * 0.05 level, paired t-test.

to predict the unobserved link. Unlike the reconstruction task, this task predicts the future links instead of reconstructing the existing links. Therefore, this task can show the performance of predictability of different network embedding methods. In addition, we add *Common Neighbor* in this task because it has been proved as an effective method to do link prediction [17].

For the first experiment, we randomly hide 15 precentage of existing links (about 4000 links) and use the *precision@k* as the evaluation metric of predicting the hidden links. We gradually increase the k from 2 to 10000 and report the result in Table 5. The best performance is highlighted in bold. Some of the observations and analysis on Table 5 are listed as follows:

- The result shows that when k increases, the performance of our method is consistently better than other network embedding methods. It demonstrates that the representations learned by our method have much better predicting power for new link formation.
- When $k = 1000$, the precision of our method is still higher than 0.9, but that of other methods quickly drops below 0.8. It demonstrates that our method can keep a high precision for links ranking ahead. Such an advantage is very important for some real-world applications such as recommendation and information retrieval, because users care more about the results ranked ahead in such applications.

In the second experiment, we change the sparsity of the networks by randomly removing a portion of links in the original network and then follow the aforementioned procedure to report the results of different network embedding methods. The result is shown in Figure 6.

The result shows that the margin between *LE* and *SDNE* or between *LE* and *LINE* becomes larger when the network is sparser. It demonstrates that the import of second-order proximity is able to make the learned representations more robust to sparse networks. Moreover, when we remove 80% of the links, our method still performs significantly better than baselines. It also demonstrates the power of *SDNE* in dealing with sparse networks.

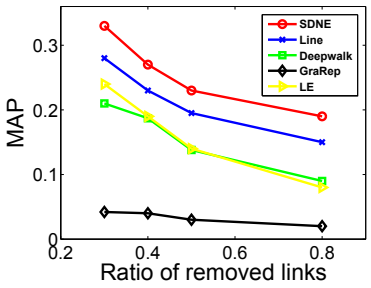
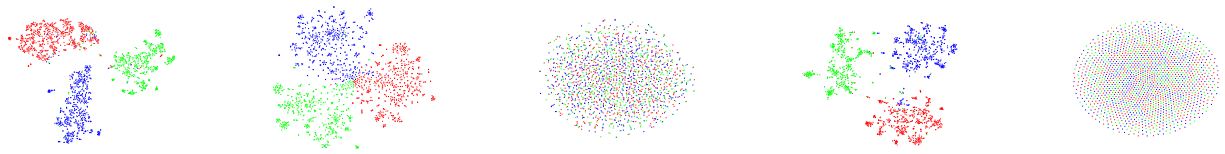


Figure 6: Performance of network embedding on networks of different sparsity. It shows that *SDNE* is more robust to the sparse network.

4.5.4 *Visualization*

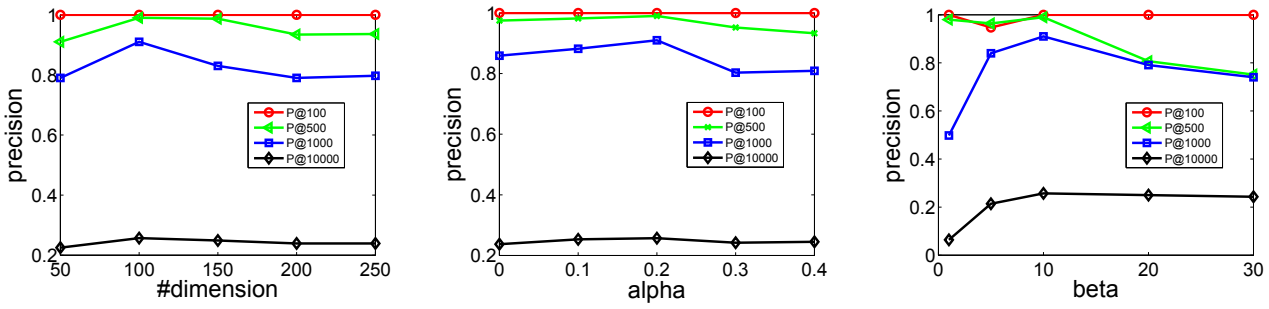
Another important application for network embedding is to generate visualizations of a network on a two-dimensional space. Therefore, we visualize the learned representations of the 20-NEWSGROUP network. We use the low-dimensional network representations learned by different network embedding methods as the input to the visualization tool *t-SNE* [31]. As a result, each newsgroup document is mapped as a two-dimensional vector. Then we can visualize each vector as a point on a two dimensional space. For documents which are labelled as different categories, we use different colors on the corresponded points. Therefore, a good visualization result is that the points of the same color are near from each other. The visualization figure is shown in Figure 7. Besides the visualization figure, similar to [4] we use the Kullback-Leibler divergence as a quantitative evaluation metric. The lower the KL divergence, the better the performance. The result is shown in Table 6.

From Figure 7, we can see that the results of *LE* and *DeepWalk* are not satisfactory because the points belonging to different categories are mixed with each other. For *LINE*, the clusters of different categories are formed. However, in the center part the documents



(a) *SDNE* (b) *LINE* (c) *DeepWalk* (d) *GraRep* (e) *LE*

Figure 7: Visualization of 20-NEWSGROUP. Each point indicates one document. Color of a point indicates the category of the document. The blue indicates the topic of rec.sport.baseball, the red indicates the topic of comp.graphics and the green indicates the topic of talk.politics.guns



(a) #dimension (b) α (c) β

Figure 8: Parameter w.r.t. the number of embedding dimensions, the value of α and the value of β

Table 6: KL-divergence for the 20-NEWSGROUP dataset

Algorithm	*SDNE*	*GraRep*	*LINE*	*DeepWalk*	*LE*
KL divergence	**0.68****	0.73	0.87	2.6	2.95

Significantly outperforms GraRep at the: ** 0.01 level.

of different categories are still mixed with each other. For *GraRep*, the result looks better because points of the same color form segmented groups. However, the boundaries of each group are not very clear. Obviously, the visualization of *SDNE* performs best in both the aspects of group separation and boundary aspects. The results shown in Table 6 also quantitatively demonstrate the superiority of our method in the visualization task.

4.6 Parameter Sensitivity

We investigate the parameter sensitivity in this section. Specifically, we evaluate how different numbers of the embedding dimensions and different values of hyper-parameter α and β can affect the results. We report $precision@k$ on the dataset of ARXIV-GRQC.

Choose an appropriate number of embedding dimensions: We first show how the dimension of the embedding vectors affects the performance in Figure 8(a). We can see that initially the performance raises when the number of dimension increases. This is intuitive as more bits can encode more useful information in the increasing bits. However, when the number of dimensions continuously increases, the performance starts to drop slowly. The reason is that too large number of dimensions may introduce noises which will deteriorate the performance. Overall, it is always important to determine the number of dimensions for the latent embedding space, but our method is not very sensitive to this parameter.

Find a good balanced point between first-order and second-order proximity: Then we show how the value of α affects the performance in Figure 8(b). The parameter of α balances the weight of the first-order proximity and second-order proximity between vertexes. When $\alpha = 0$, the performance is totally determined by the second-order proximity. And the larger the α, the more the model concentrates on the first-order proximity. From Figure 8(b), we can see that the performance of $\alpha = 0.1$ and $\alpha = 0.2$ are better than that of $\alpha = 0$. It demonstrates that both first-order and second-order proximity are essential for network embedding methods to characterize the network structure.

Concentrate more on the reconstruction error for the non-zero elements: At last, we show how the value of β affects the performance. The β controls the reconstruction weight of the non-zero elements in the training graph. The larger the β, the model will more prone to reconstruct the non-zero elements. The result is shown in Figure 8(c). We can see that when $\beta = 1$, the result is not good. Because in this case, the model will equally reconstruct the non-zero and zero elements in the training network. As we have explained before, no link between two nodes does not indicate dissimilarity of these two nodes, but the existence of a link between two nodes does indicate similarity of these two nodes. Therefore, the reconstruction of zero elements will introduce noises and thus deteriorate the performance. However, when β is too large, the performance decreases too. The reason is that, in this case, the model almost ignores the zero-elements when perform reconstruction and is prone to maintain similarity between any pair of nodes. However, many zero elements still indicate the dissimilarity between vertexes. Therefore, the performance drops. This experiment suggests that we should concentrate more on the reconstruction error of none-zero elements in the training networks but cannot totally omit the reconstruction error of zero elements when we perform network embedding.

5. CONCLUSIONS

In this paper, we propose a Structural Deep Network Embed-

ding, namely *SDNE*, to perform network embedding. Specifically, to capture the highly non-linear network structure, we design a semi-supervised deep model, which has multiple layers of non-linear functions. To further address the structure-preserving and sparsity problem, we jointly exploit the first-order proximity and second-order proximity to characterize the local and global network structure. By jointly optimizing them in the semi-supervised deep model, the learned representations are local-global structure-preserved and are robust to sparse networks. Empirically, we evaluate the generated network representations in a variety of network datasets and applications. The results demonstrate substantial gains of our method compared with state-of-the-art.

Our future work will focus on how to learn representations for a new vertex which has no linkage to existing vertexes.

6. ACKNOWLEDGMENTS

This work was supported by National Program on Key Basic Research Project, No. 2015CB352300; National Natural Science Foundation of China, No. 61370022, No. 61531006, No. 61472444 and No. 61210008. Thanks for the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

7. REFERENCES

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[2] Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[3] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[4] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015.

[5] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 119–128. ACM, 2015.

[6] N. S. Dash. Context and contextual word meaning. *SKASE Journal of Theoretical Linguistics*, 5(2):21–31, 2008.

[7] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[9] K. Georgiev and P. Nakov. A non-iid framework for collaborative filtering with restricted boltzmann machines. In *ICML-13*, pages 1148–1156, 2013.

[10] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[12] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

[13] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM, 2010.

[14] E. M. Jin, M. Girvan, and M. E. Newman. Structure of growing social networks. *Physical review E*, 64(4):046132, 2001.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[17] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[18] T. Liu and D. Tao. Classification with noisy labels by importance reweighting. *TPAMI*, (1):1–1.

[19] D. Luo, F. Nie, H. Huang, and C. H. Ding. Cauchy graph embedding. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 553–560, 2011.

[20] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[21] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710. ACM, 2014.

[22] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[23] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[24] B. Shaw and T. Jebara. Structure preserving embedding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 937–944. ACM, 2009.

[25] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[26] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[27] L. Tang and H. Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009.

[28] L. Tang and H. Liu. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1107–1116. ACM, 2009.

[29] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[30] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. Learning deep representations for graph clustering. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1293–1299, 2014.

[31] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[32] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

[33] D. Wang, P. Cui, M. Ou, and W. Zhu. Deep multimodal hashing with orthogonal regularization. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 2291–2297. AAAI Press, 2015.

[34] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[35] C. Xu, D. Tao, and C. Xu. Multi-view intact space learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(12):2531–2544, 2015.

[36] J. Zhuang, I. W. Tsang, and S. Hoi. Two-layer multiple kernel learning. In *International conference on artificial intelligence and statistics*, pages 909–917, 2011.