Learning Network Embedding with Community Structural Information

Abstract

Network embedding is an effective approach to learn the low-dimensional representations of vertices in networks, aiming to capture and preserve the structure and inherent properties of networks. The vast majority of existing network embedding methods exclusively focus on vertex proximity of networks, while ignoring the network internal community structure. However, the homophily principle indicates that vertices within the same community are more similar to each other than those from different communities, thus vertices within the same community should have similar vertex representations. Motivated by this, we propose a novel network embedding framework NECS to learn the Network Embedding with Community Structural information, which preserves the high-order proximity and incorporates the community structure in vertex representation learning. We formulate the problem into a principled optimization framework and provide an effective alternating algorithm to solve it. Extensive experimental results on several benchmark network datasets demonstrate the effectiveness of the proposed framework in various network analysis tasks including vertex classification, network reconstruction and link prediction.

1 Introduction

Networks are ubiquitous in the real world, such as social networks, bibliographic networks and communication networks. Networks often involve extensively interconnected structures, which make it difficult to directly apply machine learning algorithms on them for various network analysis tasks, such as vertex classification, network reconstruction and link prediction. Therefore, how to represent network data is a crucial problem for network analysis tasks. In recent years, network embedding has been widely recognized as an effective approach to learn the low-dimensional vector representations of vertices in networks. With proper network embedding, vector-based machine learning algorithms can be easily applied to those aforementioned network analysis tasks.

One effective strategy to obtain the low-dimensional vector representations is to preserve the proximity between vertices in networks. The adjacency matrix (i.e., first-order proximity) is the most intuitive way to reveal the network structure. However, the adjacency matrix is also very sparse and insufficient to fully characterize the relationships between vertices. In order to capture the network structure and characterize the relationships better, researchers turn their attentions to high-order proximity [Cao *et al.*, 2015; Yang *et al.*, 2017; Gao and Huang, 2018; Lian *et al.*, 2018; Zhang *et al.*, 2018]. Particularly, NEU [Yang *et al.*, 2017] significantly improves the performance of the state-of-the-art methods[Perozzi *et al.*, 2014; Tang *et al.*, 2015; Cao *et al.*, 2015; Grover and Leskovec, 2016] by approximating high-order proximity matrix.

However, the aforementioned methods predominantly focus on the vertex proximity of networks, while ignoring the community structure which is ubiquitous in networks. A community is intuitively identified as a group of vertices with more connections between its internal vertices compared with the external ones [Girvan and Newman, 2002]. The homophily principle indicates that vertices within the same community are more similar to each other than those from different communities. Network embedding aims to learn the low-dimensional vector representations of vertices while preserving both the structure and inherent properties of networks. Therefore, It is nontrivial to integrate the community structure into network embedding as community structure is an important property of network [Wang *et al.*, 2017; Cavallari *et al.*, 2017].

Inspired by the effectiveness of high-order proximity and the homophily principle, we propose a novel network embedding framework NECS to learn the Network Embedding with Community Structural information, which utilizes the high-order proximity to learn the vertex representations while incorporating the network community structure. NECS first learns the vertex representations using matrix factorization that approximates the high-order proximity. Afterwards, NECS models the community structure according to the homophily principle, and incorporates the community structure to guide the vertex representation learning. NECS optimizes the vertex representations based on both the vertex proximity together with the community structure. Thus, the vertex representations learned by NECS can preserve both the vertex proximity and the community structure of the networks, and whose effectiveness will be tested with extensive experiments on real-world benchmark network datasets. The main contributions of our paper are as follows:

- we propose a novel network embedding framework NECS, which effectively preserves the high-order vertex proximity and incorporates the community structure of networks in vertex representation learning;
- we provide an effective alternating optimization algorithm for the proposed NECS framework; and
- we evaluate the effectiveness of the proposed NECS framework on several real-world benchmark network datasets with various concrete network analysis tasks.

2 Related Work

2.1 Network Embedding

Here we give a brief introduction to existing network embedding methods. DeepWalk [Perozzi *et al.*, 2014] introduces truncated random walks to obtain the vertex sequences and adopts Skip-gram [Mikolov *et al.*, 2013] model to learn the vertex representations. LINE [Tang *et al.*, 2015] optimizes an objective function that preserves the first-order and second-order proximities during vertex representation learning. Not only the first-order and second-order proximities, but the high-order proximities are also adopted to preserve the proximity between vertices. NEU [Yang *et al.*, 2017] enhances the performance of the network embedding methods by approximating high-order proximity. INH-MF [Lian *et al.*, 2018] obtains binary code representations of information networks by preserving high-order proximity. AROPE [Zhang *et al.*, 2018] derives the vertex representations based on singular value decomposition framework while preserving arbitrary-order proximity. In addition, many other works [Cao *et al.*, 2015; Grover and Leskovec, 2016; Wang *et al.*, 2016; Gao and Huang, 2018] preserve the vertex proximity explicitly or implicitly.

2.2 Community Structure

Effectively identifying communities in a network can help us to know the network structure better and facilitate the network analysis tasks. Many types of community detection methods have been proposed to identify the communities contained in a network, such as clustering-based methods [Newman, 2004], modularity optimization methods [Newman, 2006], spectral algorithms [Tang and Liu, 2011] and so on. However, the aforementioned methods assign a vertex to only one community and lack the interpretability. In recent years, community affiliation based algorithms have attracted considerable attention [Wang *et al.*, 2011; Yang and Leskovec, 2012; Yang and Leskovec, 2013; Gligorijevic *et al.*, 2016]. Community affiliation based algorithms learn the vertex-community membership low-dimensional representations of vertices and assign vertices to the corresponding communities according to these representations. For example, NF-CCE [Gligorijevic *et al.*, 2016] decomposes adjacency matrix into a low-dimensional, nonnegative community affiliation representation matrix, and extracts communities from the matrix. By incorporating vertex representation learning with community affiliation representation learning and jointly optimizing them, we can utilize community structural information to learn more discriminative representations of vertices.

3 The Proposed Framework

In this section, we present the proposed network embedding framework - NECS in detail. We first summarize the notations used in this paper. We use bold uppercase characters for matrices(e.g., $\mathbf{A}$), and represent the (*i*, *j*)-th entry of matrix $\mathbf{A}$ as $\mathbf{A}_{ij}$, the *i*-th row of matrix $\mathbf{A}$ as $\mathbf{A}_{i*}$, the *j*-th column of matrix $\mathbf{A}$ as $\mathbf{A}_{*j}$, the transpose of $\mathbf{A}$ as $\mathbf{A}^T$, the trace of $\mathbf{A}$ as *tr*($\mathbf{A}$) if $\mathbf{A}$ is a square matrix. For an arbitrary matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, its Frobenius norm is defined as $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{d}\mathbf{A}_{ij}^2}$. In addition, $\odot$, $\mathbf{I}$, $\|\cdot\|$ and $\mathbf{1}$ denote the hadamard product, identify matrix, L2-norm of a vector and a vector with each element equals to 1, respectively.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected and unweighted graph, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of vertices and $\mathcal{E}$ is the set of edges connecting vertices in $\mathcal{V}$. Nonnegative symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ denotes the adjacency matrix of $\mathcal{G}$, where $\mathbf{A}_{ij} = 1$ if there is an edge between v_i and v_j, or $\mathbf{A}_{ij} = 0$ otherwise.

With these notations, we will discuss how to incorporate the high-order proximity and the community structure to learn the low-dimensional representations of vertices in $\mathcal{G}$ in the following subsections.

3.1 High-Order Vertex Proximity

The homophily principle indicates the correlations between vertices, which imply that the connected vertices are more likely to be similar to each other than the unconnected ones. Thus, the adjacency matrix $\mathbf{A}$ can be treated as the first-order proximity, which captures the pairwise proximity between vertices. However, the first-order proximity is also very sparse and insufficient to fully model the relationships between vertices in most cases. In order to characterize the connections between vertices better, high-order proximity is widely studied, and the order of the proximity varies considerably on different networks and target applications [Yang *et al.*, 2017; Zhang *et al.*, 2018].

Definition 1 *(High-order Proximity). Given the adjacency matrix* $\mathbf{A}$*, a high-order proximity can be defined as a polynomial function of* $\mathbf{A}$:

$$\mathbf{P} = w_1\mathbf{A} + w_2\mathbf{A}^2 + ... + w_l\mathbf{A}^l, \tag{1}$$

where l is the order, and $w_1, ..., w_l$ are the weights for each term. Matrix $\mathbf{A}^l$ denotes the *l*-order proximity matrix, which is defined as follows:

$$\mathbf{A}^l = \underbrace{\mathbf{A}...\mathbf{A}}_{l}.$$

The adjacency matrix $\mathbf{A}$ could also be replaced by normalized adjacency matrix [Yang *et al.*, 2017], Laplacian matrix, or normalized Laplacian matrix [Belkin and Niyogi, 2002].

For simplicity, we only focus on the adjacency matrix in the rest of the paper.

To integrate the high-order proximity in network embedding, the widely adopted method is nonnegative matrix factorization, which can be formulated as follows:

$$\min \|\mathbf{P}-\mathbf{V}\mathbf{U}^T\|_F^2 \quad s.t.\ \mathbf{U}\geq 0,\ \mathbf{V}\geq 0, \tag{2}$$

where the positive semi-definite matrix $\mathbf{U},\mathbf{V}\in\mathbb{R}^{n\times d}$ are the low-dimensional representations of vertices.

3.2 Community Structure

In the stochastic block model [Karrer and Newman, 2011], the value of $\mathbf{A}_{ij}$ can be viewed as the probability that there is an edge between v_i and v_j, and also determined by the probability that v_i and v_j generate edges belonging to the same community. Denote $\mathcal{C}=\{c_1,c_2,...,c_k\}$ as the set of community labels, and $\mathbf{H}=[\mathbf{H}_{ir}]\in\mathbb{R}^{n\times k}$, where $\mathbf{H}_{ir}$ represents the probability that v_i generates an edge belonging to community c_r. $\mathbf{H}_{ir}$ is also viewed as the probability of v_i belonging to community c_r, thus the probability of v_i and v_j belonging to the same community can be represented as: $\tilde{\mathbf{S}}_{ij}=\sum_{r=1}^{k}\mathbf{H}_{ir}\mathbf{H}_{jr}$. In real-world networks, the homophily principle indicates that if vertices are more similar to each other, they are more likely to belong to the same community. Thus, we can measure the similarity between two arbitrary vertices v_i and v_j as $\mathbf{S}_{ij}=\mathbf{A}_{i*}\mathbf{A}_{*j}/\|\mathbf{A}_{i*}\|\|\mathbf{A}_{*j}\|$. By minimizing the difference between $\tilde{\mathbf{S}}_{ij}$ and $\mathbf{S}_{ij}$, we can learn the community structure with the following nonnegative matrix factorization problem:

$$\min \|\mathbf{S}-\mathbf{H}\mathbf{H}^T\|_F^2 \quad s.t.\ \mathbf{H}\geq 0,\ \mathbf{H}\mathbf{1}=\mathbf{1}. \tag{3}$$

In order to utilize the community structure to learn the low-dimensional vertex representation matrix $\mathbf{U}$, we introduce a nonnegative community representation matrix $\mathbf{W}\in\mathbb{R}^{k\times d}$, where *r*-th row of $\mathbf{W}$ (e.g., $\mathbf{W}_{r*}$) is the representation of community c_r [Wang *et al.*, 2017]. Thus, with the representation $\mathbf{U}_{i*}$ of v_i, we can mathematically model the probability of v_i belonging to community c_r as $\mathbf{U}_{i*}\mathbf{W}_{r*}^T$. Without loss of generality, for all the vertices, the problem is formulated as follows:

$$\min \|\mathbf{H}-\mathbf{U}\mathbf{W}^T\|_F^2 \quad s.t.\ \mathbf{W}\geq 0, \tag{4}$$

where we learn the low-dimensional representations of vertices by minimizing the difference between $\mathbf{H}$ and $\mathbf{U}\mathbf{W}^T$. In other words, we expect to jointly optimize $\mathbf{U}$ and $\mathbf{H}$ to boost each other to obtain more discriminative representations.

With modeling the relationship between $\mathbf{U}$ and $\mathbf{H}$, the final objective function of the proposed NECS framework is formulated as follows:

$$\min_{\mathbf{V},\mathbf{U},\mathbf{W},\mathbf{H}} \|\mathbf{P}-\mathbf{V}\mathbf{U}^T\|_F^2+\alpha\|\mathbf{S}-\mathbf{H}\mathbf{H}^T\|_F^2+\beta\|\mathbf{H}-\mathbf{U}\mathbf{W}^T\|_F^2$$
$$s.t.\ \mathbf{V}\geq 0,\ \mathbf{U}\geq 0,\ \mathbf{W}\geq 0,\ \mathbf{H}\geq 0,\ \mathbf{H}\mathbf{1}=\mathbf{1}. \tag{5}$$

The problem in Eq.(5) is difficult to solve due to the constraint on $\mathbf{H}$. To tackle the issue, we relax the constraint on $\mathbf{H}$ to an orthogonal constraint instead, i.e., $\mathbf{H}^T\mathbf{H}=\mathbf{I}$ [Von Luxburg, 2007]. After the relaxation, we can further rewrite the above objective function as:

$$\begin{aligned}\min_{\mathbf{V},\mathbf{U},\mathbf{W},\mathbf{H}} &\|\mathbf{P}-\mathbf{V}\mathbf{U}^T\|_F^2+\alpha\|\mathbf{S}-\mathbf{H}\mathbf{H}^T\|_F^2\\ &+\beta\|\mathbf{H}-\mathbf{U}\mathbf{W}^T\|_F^2+\lambda\|\mathbf{H}^T\mathbf{H}-\mathbf{I}\|_F^2\\ &s.t.\ \mathbf{V}\geq 0,\ \mathbf{U}\geq 0,\ \mathbf{W}\geq 0,\ \mathbf{H}\geq 0,\end{aligned} \tag{6}$$

where we introduce a parameter λ to ensure that the orthogonal condition is satisfied. Normally, we set it as a enough large number (e.g., 10^8) to make the orthogonal condition satisfied.

4 Alternating Optimization Algorithm

The objective function of the proposed NECS framework is not jointly convex, so we cannot optimize all the variables $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{H}$ simultaneously. To optimize the objective function, we provide an alternative optimization algorithm to learn one variable while fixing others.

4.1 Update Rule for U

First, we fix $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{H}$ to update $\mathbf{U}$. Specifically, when $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{H}$ are fixed, the objective function is convex w.r.t the low-dimensional vertex representation matrix $\mathbf{U}$. Then by removing the terms that are irrelevant to $\mathbf{U}$, the objective function can be rewritten as follows:

$$\min_{\mathbf{U}\geq 0} \|\mathbf{P}-\mathbf{V}\mathbf{U}^T\|_F^2+\beta\|\mathbf{H}-\mathbf{U}\mathbf{W}^T\|_F^2. \tag{7}$$

The Lagrangian function of Eq.(7) is:

$$\mathcal{L}(\mathbf{U})=\|\mathbf{P}-\mathbf{V}\mathbf{U}^T\|_F^2+\beta\|\mathbf{H}-\mathbf{U}\mathbf{W}^T\|_F^2-tr(\mathbf{\Lambda}_{\mathrm{U}}\mathbf{U}^T), \tag{8}$$

where $\mathbf{\Lambda}_{\mathrm{U}}$ is the Lagrange multiplier for the constraint $\mathbf{U}\geq 0$. By taking the derivative of $\mathcal{L}(\mathbf{U})$ w.r.t. $\mathbf{U}$ and setting it to zero, we can get:

$$\mathbf{\Lambda}_{\mathrm{U}}=-\mathbf{P}^T\mathbf{V}+\mathbf{U}\mathbf{V}^T\mathbf{V}-\beta\mathbf{H}\mathbf{W}+\beta\mathbf{U}\mathbf{W}^T\mathbf{W}. \tag{9}$$

With the KKT complementary condition for the nonnegativity of $\mathbf{U}$, i.e., $[\mathbf{\Lambda}_{\mathrm{U}}]_{ij}\mathbf{U}_{ij}=0$, we have:

$$[-\mathbf{P}^T\mathbf{V}+\mathbf{U}\mathbf{V}^T\mathbf{V}-\beta\mathbf{H}\mathbf{W}+\beta\mathbf{U}\mathbf{W}^T\mathbf{W}]_{ij}\mathbf{U}_{ij}=0, \tag{10}$$

which leads to the following update rule for $\mathbf{U}$:

$$\mathbf{U}_{ij}\leftarrow\mathbf{U}_{ij}\sqrt{\frac{[\mathbf{P}^T\mathbf{V}+\beta\mathbf{H}\mathbf{W}]_{ij}}{[\mathbf{U}\mathbf{V}^T\mathbf{V}+\beta\mathbf{U}\mathbf{W}^T\mathbf{W}]_{ij}}}. \tag{11}$$

4.2 Update Rules for V, W and H

Similar to the process of getting the update rule for $\mathbf{U}$, we can update $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{H}$ by the following update rules:

$$\mathbf{V}_{ij}\leftarrow\mathbf{V}_{ij}\sqrt{\frac{[\mathbf{P}\mathbf{U}]_{ij}}{[\mathbf{V}\mathbf{U}^{\mathbf{T}}\mathbf{U}]_{ij}}}, \tag{12}$$

$$\mathbf{W}_{ij}\leftarrow\mathbf{W}_{ij}\sqrt{\frac{[\mathbf{H}^{\mathbf{T}}\mathbf{U}]_{ij}}{[\mathbf{W}\mathbf{U}^{\mathbf{T}}\mathbf{U}]_{ij}}}, \tag{13}$$

$$\mathbf{H}_{ij}\leftarrow\mathbf{H}_{ij}\sqrt{\frac{\sqrt{\mathbf{\Phi}_{ij}}}{2(\alpha+\lambda)[\mathbf{H}\mathbf{H}^T\mathbf{H}]_{ij}}}, \tag{14}$$

where $\mathbf{\Phi}=(\alpha+\lambda)\mathbf{H}\mathbf{H}^T\mathbf{H}\odot(4\alpha\mathbf{S}\mathbf{H}+2\beta\mathbf{U}\mathbf{W}^T+(4\lambda-2\beta)\mathbf{H})$. Such an updating procedure will continue until convergence.

4.3 Framework Learning Analysis

In each iteration, we update $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{H}$ iteratively. Following the definitions and lemmas in [Lee and Seung, 2001; Wang *et al.*, 2011], we can define auxiliary functions $\mathcal{F}(\theta, \tilde{\theta})$ for θ, where $\theta \in \{\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{H}\}$. By minimizing these auxiliary functions, we can get the corresponding update rules, which hold the following inequality condition:

$$\begin{aligned}
&\mathcal{J}((\mathbf{U})_{t+1}, (\mathbf{V})_{t+1}, (\mathbf{W})_{t+1}, (\mathbf{H})_{t+1}) \\
\leq &\mathcal{J}((\mathbf{U})_{t+1}, (\mathbf{V})_{t+1}, (\mathbf{W})_{t+1}, (\mathbf{H})_{t}) \\
\leq &\mathcal{J}((\mathbf{U})_{t+1}, (\mathbf{V})_{t+1}, (\mathbf{W})_{t}, (\mathbf{H})_{t}) \\
\leq &\mathcal{J}((\mathbf{U})_{t+1}, (\mathbf{V})_{t}, (\mathbf{W})_{t}, (\mathbf{H})_{t}) \\
\leq &\mathcal{J}((\mathbf{U})_{t}, (\mathbf{V})_{t}, (\mathbf{W})_{t}, (\mathbf{H})_{t})
\end{aligned} \tag{15}$$

which proves the convergence of the proposed framework.

Based on the descriptions aforementioned, the main computation of NECS is in computing the update values for $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{H}$ in each iteration. The computation cost of updating $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{H}$ are $\mathcal{O}(nd^2 + n^2d + d^2k)$, $\mathcal{O}(n^2k)$, $\mathcal{O}(ndk)$ and $\mathcal{O}(n^2d + n^2k)$, respectively. Thus, the overall computation cost of NECS is $\mathcal{O}(n^2d + n^2k)$, which is in the same order of magnitude as nonnegative matrix factorization.

5 Experiments

5.1 Datasets and Baseline Methods

To verify the effectiveness of the proposed framework, we conduct experiments on the following eleven network datasets, including WebKB(Cornell, Texas, Washington, Wisconsin)[1] [Wang *et al.*, 2017], Citeseer[1] [McCallum *et al.*, 2000], Cora[1] [McCallum *et al.*, 2000], Polbooks[2], Football[2], Polblogs[2] [Adamic and Glance, 2005], Wiki[3] and Email[4]. The detailed statistics of these datasets are summarized in Table 1.

We compare the proposed framework with the following state-of-the-art methods:

- Deepwalk [Perozzi *et al.*, 2014]: It adopts random walk and skip-gram model to learn the vertex representations.
- LINE [Tang *et al.*, 2015]: It optimizes an objective function that preserves the first-order and second-order proximities to learn the vertex representations. LINE also achieves a better performance when preserving both the first-order and second-order proximities by concatenating the corresponding vertex representations.
- Node2Vec [Grover and Leskovec, 2016]: It obtains the vertex representations by maximizing the likelihood of preserving network neighborhoods of vertices.
- GraRep [Cao *et al.*, 2015]: It integrates global structural information to generate the vertex representations.
- SDNE [Wang *et al.*, 2016]: It exploits both the first-order and second-order proximities jointly in a semi-supervised deep model to generate the vertex representations.

[1]http://www.cs.umd.edu/%7Esen/lbc-proj/LBC.html

[2]http://www-personal.umich.edu/%7Emejn/netdata/

[3]https://github.com/thunlp/MMDW/tree/master/data

[4]http://snap.stanford.edu/data/email-Eu-core.html

- M-NMF [Wang *et al.*, 2017]: It incorporates vertex proximity and modularity-based community detection problem to learn the vertex representations.

For a fair comparison, we set the representation dimension *d* as 128 for all methods. For DeepWalk, we set the window-size as 5, the number of walks as 10 and the walk-length as 40. For LINE, we set the number of negative samples as 5, the total number of training samples as 10 million and the starting learning rate as 0.025. For Node2Vec, we set the number of walks as 10, the walk-length as 80, p as 1, q as 1 and the window size as 5. For GraRep, we set the maximum matrix transition as 4. For SDNE, we set α as 100, β as 10 and batch size as 16. For M-NMF, we tune α and β from {0.1, 0.5, 1, 5, 10} to get the best performance. In NECS, we vary the order *l* from {1, 2, 3} and α, β from {0.1, 0.2, 0.5, 1, 2, 5, 10}. To shrink the searching space for hyper-parameters of high-order proximity, we simply fix the weights as $w_i = \mu^{i-1}$ and vary μ from {0.1, 0.2, 0.3}.

5.2 Vertex Classification

For vertex classification task, we randomly sample 80% vertices as training set and the remaining vertices as test set, and then adopt the L2-regularized logistic regression implemented by LIBLINEAR[Fan *et al.*, 2008] to build the classifiers. We conduct experiments on all the eleven datasets described in Table 1 and repeat the process 5 times. The average accuracies are reported in Table 2.

We use NECS0 to denote the formulation defined in Eq.(2), and note that NECS0 achieves a comparable performance, which verifies the effectiveness of the vertex proximity $\mathbf{P}$. The results show that NECS outperforms the baselines on the majority of datasets. Especially on WebKB, Polbooks and Football, NECS achieves a significant improvement. Although NECS achieves the second and third best results on Wiki and Cora respectively, the results are evidently better than the remaining baselines. Comparing NECS with NECS0, NECS achieves a significant improvement over NECS0 on all the datasets, which demonstrates the necessity and effectiveness of incorporating network embedding and the community structure.

5.3 Network Reconstruction

The capability of reconstructing the original network is a basic evaluation metric for network embedding, and the vertex representations learned by a good network embedding method should preserve the original network structure. Generally, a larger similarity between two vertices implies that they are more likely to be linked. Thus, we rank pairs of vertices according to their similarities, i.e. the inner product of vertex representations. In this section, we use Citeseer, Cora, Wiki and Polblogs as representatives to assess the performance of network reconstruction in term of *precision@K*, which is defined as follows:

$$precision@K = \frac{|\{e_{ij} | e_{ij} \in \mathcal{E}_p \cap \mathcal{E}_o\}|}{|\mathcal{E}_p|}$$

where $\mathcal{E}_p$ is the set of top K predicted links, $\mathcal{E}_o$ is the observed links and $|\cdot|$ represents the size of a set.

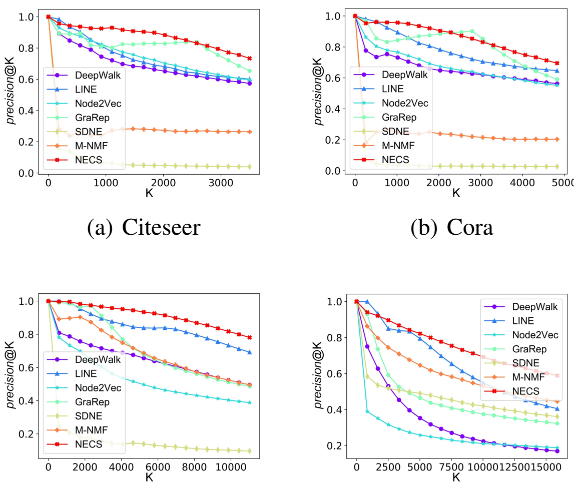
Datasets	Cornell	Texas	Washington	Wisconsin	Polbooks	Football	Citeseer	Cora	Wiki	Email	Polblogs
$\lvert\mathcal{V}\rvert$	183	183	215	251	105	115	2110	2485	2357	986	1222
$\lvert\mathcal{E}\rvert$	277	279	365	450	441	613	3668	5069	11592	16064	16714
Avg. degree	1.51	1.52	1.70	1.79	4.20	5.33	1.74	2.04	4.92	16.29	13.68
No. of labels	5	5	5	5	3	12	6	7	17	42	2

Table 1: Statistics of Datasets.

Methods	Cornell	Texas	Washington	Wisconsin	Polbooks	Football	Citeseer	Cora	Wiki	Email	Polblogs
DeepWalk	37.19	49.84	49.21	46.98	84.95	87.83	73.57	84.68	**68.74**	76.34	95.15
LINE	27.03	43.24	35.81	38.43	72.00	90.78	62.05	77.28	60.66	71.78	94.92
Node2Vec	37.51	55.16	54.80	48.31	87.81	41.57	74.74	**85.03**	65.04	63.72	95.84
GraRep	54.59	69.30	58.70	58.43	81.33	90.43	71.27	81.19	65.29	73.66	95.05
SDNE	45.08	61.95	54.79	50.59	83.24	74.43	41.61	49.58	47.19	56.10	91.80
M-NMF	46.27	66.92	64.47	58.35	88.86	89.91	73.06	81.15	65.08	74.59	95.90
NECS0	51.78	69.19	63.35	58.20	85.71	85.57	74.14	82.22	61.87	72.57	95.71
NECS	**60.54**	**73.51**	**70.69**	**66.27**	**91.43**	**96.52**	**76.16**	84.59	68.18	**79.90**	**96.49**

Table 2: Vertex classification accuracies(%).

Figure 1 shows the performance of network reconstruction with different K, and the maximum K equals to the number of the edges $|\mathcal{E}|$. On Wiki, NECS outperforms all the baselines significantly. On the remaining datasets, LINE performs better than NECS when K is relatively small, and NECS begins to generally outperform the baselines when K becomes relatively large. A point to note is that when K varies in the range of 2500 to 3000 on Cora, the precision scores of GraRep are higher than NECS. A possible reason is that GraRep integrates the network structural information to learn the vertex representations, which demonstrates the necessity of the integrating network structural information into network embedding. And the same goes for the network community structural information.



(a) Citeseer (b) Cora

(c) Wiki (d) Polblogs

Figure 1: Network reconstruction results in terms of *precision@K*.

5.4 Link Prediction

For link prediction task, we randomly remove 10% edges as test set for evaluation and utilize the remaining edges to learn the vertex representations. Similar to the network reconstruction task, we rank pairs of vertices in a similar way as network reconstruction and conduct experiments on Citeseer, Cora, Wiki and Polblogs to assess the performance of link prediction in terms of *precision@K* and *map@K*. The formula of *map@K* is:

$$AP@K(i) = \frac{\sum_{j=1}^{K} precision_i(j) \cdot \triangle_i(j)}{|\{\triangle_i(j) = 1\}|}$$
$$map@K = \frac{\sum_{v_i \in \mathcal{V}_t} AP(i)}{|\mathcal{V}_t|}$$

where $\mathcal{V}_t$ is the set of vertices whose edges are removed, and $\triangle_i(j) = 1$ indicates v_i and v_j have an edge.

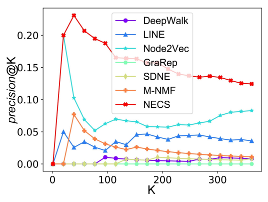
Figure 2 shows the *precision@K* values with different K, and the maximum K equals to the number of the removed edges. The results show that NECS outperforms all the baselines significantly, especially when K equals to the number of the removed edges, NECS achieves at least 40% improvement over all the baselines. The precision scores for all the methods are lower on Citeseer than other datasets, this is mainly because Citeseer is relatively sparser, which makes it difficult to predict links. But NECS still achieves significantly better results in that case. Furthermore, Table 3 shows the $map@K$ values with different K. We can see that NECS outperforms the baselines on all the datasets except Cora. NECS achieves the second best results on $map@20$, $map@50$ and $map@100$ on Cora. The performance of NECS in link prediction task demonstrates its ability to capture the network structure.

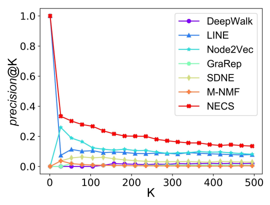
5.5 Parameter Study

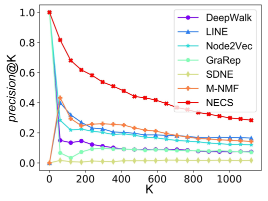
NECS has five parameters α, β, μ, l and k. Among them, α and β control how well the network embedding preserves the

Methods	Citeseer			Cora			Wiki			Polblogs		
map@K	20	50	100	20	50	100	20	50	100	20	50	100
DeepWalk	0.036	0.030	0.022	0.036	0.028	0.020	0.048	0.037	0.027	0.004	0.006	0.008
LINE	0.056	0.041	0.028	0.046	0.032	0.022	0.056	0.042	0.030	0.009	0.010	0.010
Node2Vec	0.060	0.043	0.030	**0.051**	**0.035**	**0.024**	0.050	0.038	0.027	0.024	0.023	0.020
GraRep	0.037	0.033	0.024	0.033	0.026	0.019	0.050	0.040	0.029	0.027	0.026	0.024
SDNE	0.013	0.009	0.007	0.011	0.008	0.006	0.012	0.010	0.008	0.043	0.033	0.026
M-NMF	0.042	0.037	0.027	0.027	0.023	0.018	0.050	0.038	0.028	0.027	0.027	0.026
NECS	**0.061**	**0.044**	**0.031**	0.046	0.032	0.023	**0.059**	**0.044**	**0.032**	**0.048**	**0.041**	**0.035**

Table 3: Link prediction results in terms of *map@K*.


(a) Citeseer


(b) Cora


(c) Wiki

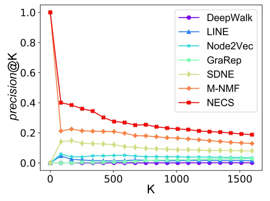
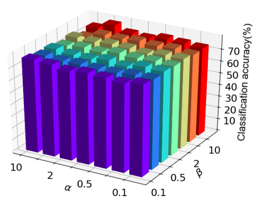
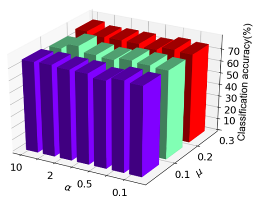

(d) Polblogs

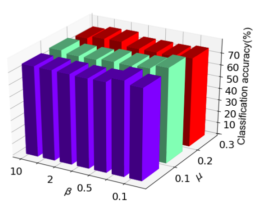
Figure 2: Link prediction results in terms of *precision@K*.

community structure, μ and l control to what extent the high-order proximity contributes to vertex proximity, and k controls the number of communities we desire to extract. For the number of communities k, we just simply set k as the known number of communities. Generally speaking, we can use the method in [Riolo *et al.*, 2017] to estimate k. We further investigate the impacts of parameters of α, β, μ and l on the performance of NECS using vertex classification task as an example. Due to the space limit, we only show the parameter study results on the Email dataset as we have similar observations on other datasets. To study how its variation affects the performance, we fix one parameter each time and vary the others. The performance variation of these parameters are shown in Figure 3(a)-(c). Particularly, we set l = 2 so as to plot in a 3-D figure. We can see that the performance is relatively stable, however, without clear trends. This is mainly because α and β collectively control the degree of network embedding preserving the community structure.

To show the convergence of NECS intuitively, the value of the loss function in Eq.(6) at the end of each iteration is recorded and normalized by $|\mathcal{V}|$ for different datasets so as to illustrate them in one figure. For simplicity, we set α, β, μ and l as 1, 1, 0.1 and 2 for all the datasets, respectively. As illustrated in Figure 3(d), NECS converges very quickly.


(a) Effect of α and β(μ=0.2)


(b) Effect of α and μ(β=10)


(c) Effect of β and μ(α=5.0)

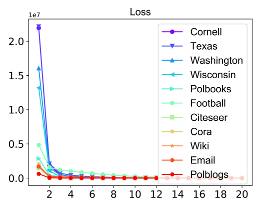

(d) Convergence of NECS

Figure 3: Parameter sensitivity and convergence.

6 Conclusion

In this paper, we propose a novel network embedding framework NECS, which preserves the high-order proximity and incorporates the community structure. First, we use nonnegative matrix factorization to learn the vertex representations by preserving the high-order proximity. Then, we propose to obtain the community structure by approximating the similarity between vertices according to the homophily principle. Finally, we establish the consensus relationship between the vertex representations and the community structure and jointly optimize them. The cooperation of the vertex representations and the community structure can boost each other to obtain more discriminative representations. Methodologically, we provide an alternating optimizing algorithm for the proposed framework. Extensive experimental results demonstrate the effectiveness of the proposed framework in several network analysis tasks. In the future, we strive to generalize this framework to signed networks.

References

[Adamic and Glance, 2005] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

[Belkin and Niyogi, 2002] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of NIPS*, pages 585–591, 2002.

[Cao *et al.*, 2015] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of CIKM*, pages 891–900, 2015.

[Cavallari *et al.*, 2017] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of CIKM*, pages 377–386, 2017.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[Gao and Huang, 2018] Hongchang Gao and Heng Huang. Self-paced network embedding. In *Proceedings of KDD*, pages 1406–1415, 2018.

[Girvan and Newman, 2002] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[Gligorijevic *et al.*, 2016] Vladimir Gligorijevic, Yannis Panagakis, and Stefanos P. Zafeiriou. Non-negative matrix factorizations for multiplex network analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2016.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of KDD*, pages 855–864, 2016.

[Karrer and Newman, 2011] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.

[Lee and Seung, 2001] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of NIPS*, pages 556–562, 2001.

[Lian *et al.*, 2018] Defu Lian, Kai Zheng, Vincent W Zheng, Yong Ge, Longbing Cao, Ivor W Tsang, and Xing Xie. High-order proximity preserving information network hashing. In *Proceedings of KDD*, pages 1744–1753, 2018.

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

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, 2013.

[Newman, 2004] Mark EJ Newman. Detecting community structure in networks. *The European Physical Journal B*, 38(2):321–330, 2004.

[Newman, 2006] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of KDD*, pages 701–710, 2014.

[Riolo *et al.*, 2017] Maria A Riolo, George T Cantwell, Gesine Reinert, and Mark EJ Newman. Efficient method for estimating the number of communities in a network. *Physical Review E*, 96(3):032310, 2017.

[Tang and Liu, 2011] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of WWW*, pages 1067–1077, 2015.

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

[Wang *et al.*, 2011] Fei Wang, Tao Li, Xin Wang, Shenghuo Zhu, and Chris Ding. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, 22(3):493–521, 2011.

[Wang *et al.*, 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of KDD*, pages 1225–1234, 2016.

[Wang *et al.*, 2017] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Proceedings of AAAI*, 2017.

[Yang and Leskovec, 2012] Jaewon Yang and Jure Leskovec. Community-affiliation graph model for overlapping network community detection. In *Proceedings of ICDM*, pages 1170–1175, 2012.

[Yang and Leskovec, 2013] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of WSDM*, pages 587–596, 2013.

[Yang *et al.*, 2017] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. Fast network embedding enhancement via high order proximity approximation. In *Proceedings of IJCAI*, pages 19–25, 2017.

[Zhang *et al.*, 2018] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. Arbitrary-order proximity preserved network embedding. In *Proceedings of KDD*, pages 2778–2786, 2018.