2016 IEEE 16th International Conference on Data Mining

Homophily, Structure, and Content Augmented Network Representation Learning

Daokun Zhang*, Jie Yin†, Xingquan Zhu‡§, and Chengqi Zhang*
*Center for Quantum Computation & Intelligent Systems, FEIT, University of Technology, Sydney, Australia
Email: Daokun.Zhang@student.uts.edu.au, Chengqi.Zhang@uts.edu.au
†CSIRO, Sydney, Australia
Email: Jie.Yin@csiro.au
‡Department of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, USA
§School of Computer Science, Fudan University, China
Email: xqzhu@cse.fau.edu

***Abstract*—Advances in social networking and communication technologies have witnessed an increasing number of applications where data is not only characterized by rich content information, but also connected with complex relationships representing social roles and dependencies between individuals. To enable knowledge discovery from such networked data, network representation learning (NRL) aims to learn vector representations for network nodes, such that off-the-shelf machine learning algorithms can be directly applied. To date, existing NRL methods either primarily focus on network structure or simply combine node content and topology for learning. We argue that in information networks, information is mainly originated from three sources: (1) homophily, (2) topology structure, and (3) node content. Homophily states social phenomenon where individuals sharing similar attributes (content) tend to be directly connected through local relational ties, while topology structure emphasizes more on global connections. To ensure effective network representation learning, we propose to augment three information sources into one learning objective function, so that the interplay roles between three parties are enforced by requiring the learned network representations (1) being consistent with node content and topology structure, and also (2) following the social homophily constraints in the learned space. Experiments on multi-class node classification demonstrate that the representations learned by the proposed method consistently outperform state-of-the-art NRL methods, especially for very sparsely labeled networks.**

I. Introduction

Recent years have witnessed an enormous growth of information networks, such as social networks, communication networks, and citation networks, generated from all aspects of social applications. Unlike conventional vector-based data, instances in information networks are neither independent nor identically distributed, but are connected to each other with complex dependencies. To process such networked data, one critical problem is to accurately learn a useful network representation, which aims at embedding a network into a low-dimensional space, *i.e.*, learning a vector representation for each node, with the objective of preserving network structure in the embedded space. As a result, off-the-shelf vector-based machine learning techniques can be easily applied. Existing studies on network representation learning (NRL) [1, 2] have shown that NRL can effectively capture network structure to facilitate subsequent analytic tasks, such as node classification [3], anomaly detection [4], and link prediction [5].

Despite its great potential, learning useful network representations faces several challenges. (1) *Sparsity*: Real-world networks are often sparse and have very few observed links, due to reasons such as privacy concerns or legal restrictions. This makes it very difficult to discover hidden relatedness between nodes that are not explicitly connected. (2) *Structure-preserving*: NRL has the goal of preserving network structure in the learned space. However, because the underlying network structure is very complex, the similarity between nodes depends on both the local and global network structure, such as the local social roles of individuals and the broader community connections. Thus, simultaneously preserving the local and global structure is a challenging task. (3) *Rich-Content*: Besides structure information, many real-world networks often contain rich content on node attributes. Such content information is helpful or even critical in some tasks such as node classification. How to leverage rich content information and its interplay with network structure for representation learning remains an open problem.

To tackle these challenges, we consider three information sources for learning useful network representations.

Homophily (or the first-order proximity): Homophily phenomenon [6] is a known fact in social networks where nodes with similar attributes (content) tend to be connected through relational ties, which contribute to local structure of individual nodes [7]. To preserve such local connectivity in the network, nodes directly connected to each other should be embedded closely together in the learned representation space.

Structural Context: In addition to direct neighbors, nodes in a network also have dependencies with indirect neighbors. To respect the global network structure, nodes sharing common neighbors (*i.e.*, context nodes) should be close to each other in the learned space. Structural context includes the second-order proximity and higher order proximity. Different from homophily, structural context does not focus on local connectivity but emphasizes on structural roles of nodes in broader connections, so nodes can be far apart from each other in a

2374-8486/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDM.2016.139

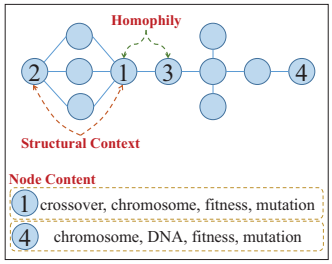


Fig. 1. A toy example of information network. Our proposed HSCA algorithm utilizes all three information sources (*i.e.*, homophily, structural context, and node content), for learning useful network representations.

network but still share similar structural roles.

Node Content: Rich information on node attributes provides the most direct evidence to measure content-based similarity of nodes. Thus, nodes having similar content features should also be represented closely to each other in the new space.

Fig. 1 gives an illustrative example. As node 1 and node 3 are directly connected, they should be represented closely to each other in the embedded space, to reflect the homophily principle. On the other hand, though there is no direct link between node 1 and node 2, they share three common neighbors, *i.e.*, they share the same structural context. Thus, they should also be represented closely to each other. Furthermore, although node 1 and node 4 are not directly connected nor share common neighbors, they are likely to have similar representations because they share similar content on node attributes. We expect that the consideration of all three information sources (*i.e.*, homophily, contextual structure, and node content), and more importantly, the interplay between three parties can maximally contribute to learning effective network representations.

To date, most of existing work on NRL (*e.g.*, [1, 2, 8]) learns representations from network structure only. For example, DeepWalk [1] generalizes the Skip-Gram model [9], which learns word representations using word context in sentences, by taking random walk sequences in graphs as the equivalence of sentences. As an analogy to word context, nodes with similar structural context in random walk sequences tend to have similar representations. Hence, only structural context is utilized by DeepWalk to learn node representations. LINE [2] learns node representations by explicitly modeling two kinds of pairwise node relationships, *i.e.* the first-order proximity and the second-order proximity. GraRep [10] moves a further step to consider different k-step proximity (in particular when $k > 2$), for forming a global representation. However, both LINE and GraRep only consider homophily and structural context in the original network space, but not in the new representation space. In addition, both methods do not consider node content information for network representation learning. Only very recently, Text-Associated DeepWalk (TADW) [11] attempts to import node features into matrix factorization based DeepWalk. TADW inherits the information source of structural context from DeepWalk and exploits node content for improving representation learning. Table I summarizes the information sources used by all these methods. Clearly, none of these methods have utilized all above three information sources to learn informative node representations.

TABLE I
A SUMMARY OF NRL ALGORITHMS BASED ON THE INFORMATION SOURCES USED.

Algorithm	Homophily	Structural Context	Node Content
DeepWalk [1]		✓	
LINE [2]	✓	✓	
GraRep [10]	✓	✓	
TADW [11]		✓	✓
HSCA (our algorithm)	✓	✓	✓

In this paper, to fill this gap, we propose a new NRL algorithm, called HSCA (Homophily, Structure, and Content Augmented network representation learning), that effectively learns a latent network representation using all three information sources. Inspired by TADW [11], we formulate the problem of network representation learning as a new matrix factorization problem that 1) considers broader structural context in random walk sequences to discover hidden relatedness between nodes; 2) imports nodes' content features into the matrix factorization process; and 3) introduces a regularization term to enforce homophily between directly connected nodes in the new representation space. To this end, we design a new optimization objective function and derive an efficient algorithm based on conjugate gradient methods to solve it. By simultaneously integrating homophily, structural context, and node content, HSCA effectively embeds a network into a single latent representation space that captures the interplay between the three information sources. As a result, the representations learned by HSCA have the advantage of not only preserving network structure and node content information, but also explicitly following the homophily constraints in the new space. We evaluate the performance of the HSCA algorithm on real-world networks for the task of multi-class node classification. Experimental results show that, compared with the state-of-the-art NRL algorithms, HSCA's representations achieve better classification performance, especially for sparsely labeled networks.

The remainder of the paper is organized as follows. Section II reviews previous work related to NRL. The problem definition and background work directly related to our formulation are given in Section III, followed by the proposed algorithm described in Section IV. Experiments and discussions are presented in Section V. Finally, this paper is concluded in Section VI.

II. Related Work

In this section, we review two branches of related work including graph embedding and network presentation learning.

A. Graph Embedding

Our work is related to traditional graph embedding methods, such as multidimensional scaling (MDS) [12], IsoMap [13],

locally linear embedding (LLE) [14], and Laplacian Eigenmap [15]. Given a set of vector-based instances, these methods typically first construct an affinity graph, *e.g.*, the k-nearest neighbor graph, using the similarity or Euclidean distance between pairwise data points, and then embed the affinity graph into a low-dimensional space. There is a clear distinction between the aims of research work on graph embedding and NRL. Graph embedding aims to reduce the dimensionality of the original data in the vector-based form, while NRL seeks to learn effective vector representations of network nodes to facilitate downstream network analytic tasks.

B. Network Representation Learning

Network representation learning (NRL) mines useful information from a network to learn a vector representation for each node, so that nodes close to each other in the original network space also have similar latent representations in the learned space.

Earlier work on NRL has attempted to partition a graph into different communities for creating social dimensions [8, 16, 17]. These methods usually cluster nodes into different affiliations. The representation for each node is determined by its membership with respect to different affiliations. Clustering algorithms used for this purpose include spectral clustering [18], modularity maximization [19], and EdgeCluster [17]. These studies provide a cluster-centric view of the network, but ignore important structure information of individual nodes.

Recently, several algorithms have been proposed for network representation learning [1, 2, 8]. DeepWalk [1] borrows the idea of the Skip-Gram model [9], originally designed for learning word representations using word context in sentences, to learn the representations of nodes in a network. In DeepWalk, short node sequences generated by random walks are taken as sentences, and vertex representations are learned using contextual information in random walk sequences. DeepWalk has been shown to outperform earlier NRL methods for creating social dimensions in multi-label classification tasks.

Another recently proposed work is LINE [2], which explicitly defines a loss function to capture two types of node relationships, *i.e.* the first-order proximity that nodes linked to each other tend to have similar latent representations, and the second-order proximity that nodes sharing direct common neighbors tend to have similar latent representations. LINE introduces two different models to capture 1-step and 2-step node relationships, and combines node representations learned by the two models as the final representation. GraRep [10] steps forward to consider different k-step proximity (in particular for $k > 2$), for learning network representations. However, both LINE and GraRep learn representations separately for k-step proximity and simply concatenate the learned vector features to form the final network representation. Such a simple combination inherently neglects the interplay between different types of proximity, including their relative importance, possibly leading to unsatisfactory performance.

All of the above methods have focused on utilizing network structure only to learn network representations. Only very recently, Text-Associated DeepWalk (TADW) [11] is proposed to incorporate text features of nodes into network representation learning. In TADW, DeepWalk is proved to be equivalent to matrix factorization. Based on this formulation, text features of nodes are naturally encoded into the matrix factorization framework. The output of matrix factorization is then taken as the learned vertex representations.

Motivated by TADW, our work proposes to explicitly enforce the homophily property of connected nodes in the learned representation space so as to learn an effective network representation. By simultaneously augmenting homophily, structural context, and node content, the representations learned by our algorithm can better capture the interplay between node content information and network structure that best benefit network representation learning.

III. Problem Definition and Preliminaries

In this section, we formally define our NRL problem and review preliminaries of Text-Associated DeepWalk (TADW).

A. Problem Definition

Given an undirected graph $G = (\mathcal{V}, \mathcal{E}, T)$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. T is the node text feature matrix with the i-th column of T, $\boldsymbol{t}_i$, characterizing node $\boldsymbol{v}_i \in \mathcal{V}$. Our problem **aims** to embed graph G into a low-dimensional space, *i.e.*, learning a vector representation $\boldsymbol{x} \in \mathbb{R}^k$ for each node $\boldsymbol{v} \in \mathcal{V}$ in G.

The learned representation $\boldsymbol{x}$ should satisfy two properties: (1) low-dimensionality, the dimension k of the embedded space is expected to be much smaller than $|\mathcal{V}|$, and (2) informativeness, the learned representation $\boldsymbol{x}$ preserves the graph structure information reflected in $\mathcal{E}$ and node content information in node text features T. With these two properties, the learned representation can be taken as input to subsequent network mining tasks, like node classification.

B. Text-Associated DeepWalk

Inspired by DeepWalk [1], Text-Associated DeepWalk (TADW) [11] incorporates node text features into network representation learning. In DeepWalk, nodes in a network are taken as words in natural language and a set of short random walk sequences are generated from the given network to provide context for nodes. Given a node v_i and its context nodes within t window size, $\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i$, the representation for v_i, $\Phi(v_i)$, is learned by maximizing the conditional probability:

$$\mathrm{P}(\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i | \Phi(v_i)) = \prod_{j=i-t, j\neq i}^{i+t} \mathrm{P}(v_j | \Phi(v_i)). \quad (1)$$

In [1], the probability $\mathrm{P}(v_j|\Phi(v_i))$ is approximated using Hierarchical Softmax [20] to speed up training.

We denote the PageRank matrix of graph G as $P \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. The degree of node v_i is denoted by d_i. We then have $P_{ij} = 1/d_i$, if $(v_i, v_j) \in \mathcal{E}$; otherwise, $P_{ij} = 0$. In [11],

DeepWalk is proved to be equivalent to factorizing a matrix $M \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ given by

$$M_{ij} = \log \frac{\left[P + P^2 + \cdots + P^t\right]_{ij}}{t}. \tag{2}$$

Matrix M can be factorized as two low-rank matrices $W \in \mathbb{R}^{k\times|\mathcal{V}|}$ and $H \in \mathbb{R}^{k\times|\mathcal{V}|}$ by solving the following problem

$$\min_{W,H} \frac{1}{2}\|M - W^{\mathrm{T}}H\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2). \tag{3}$$

The first minimization term guarantees precise factorization. The minimization of $\|W\|_F^2$ and $\|H\|_F^2$ imposes low-rank constraint on W and H, and $\frac{\lambda}{2}$ controls the trade-off between these two terms. After this matrix factorization problem is solved, W and H can be concatenated together to form a $2k$-dimensional representation for each node.

To make the representations learned by matrix factorization based DeepWalk more informative, TADW imports node text features $T \in \mathbb{R}^{f_t\times|\mathcal{V}|}$ into the matrix factorization of (3). The matrix factorization formulation is changed into

$$\min_{W,H} \frac{1}{2}\|M - W^{\mathrm{T}}HT\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2), \tag{4}$$

where $H \in \mathbb{R}^{k\times f_t}$. For computational efficiency, in (4), M is approximated by $(P + P^2)/2$. After W and H are solved, node representations are generated by concatenating W and HT as feature vectors.

IV. HSCA: Homophily, Structure, and Content Augmented Network Representation Learning

In this section, we formulate the optimization problem of our proposed method and elaborate solutions to this problem.

A. The Optimization Problem

TADW can learn more informative representations than DeepWalk, by importing text features into the matrix factorization formulation of DeepWalk. However, TADW's representations lack the ability to preserve the homophily property of the network. Thus, we propose to enforce homophily between connected nodes in our matrix factorization formulation.

By solving the problem of (4), the i-th node is then represented by $\left[\boldsymbol{w}_i^{\mathrm{T}}, (H\boldsymbol{t}_i)^{\mathrm{T}}\right]^{\mathrm{T}}$. To make connected nodes close enough to each other in the learned $2k$-dimensional space, the distance between their feature vectors should be constrained in the representation space. To achieve this, we import a regularization term $\mathcal{R}(W,H)$ into the minimization problem of (4). The regularization term is defined as

$$\begin{aligned}\mathcal{R}(W,H) &= \frac{1}{4}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}\| \begin{bmatrix}\boldsymbol{w}_i \\ H\boldsymbol{t}_i\end{bmatrix} - \begin{bmatrix}\boldsymbol{w}_j \\ H\boldsymbol{t}_j\end{bmatrix} \|_2^2 \\ &= \frac{1}{4}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}\|\boldsymbol{w}_i - \boldsymbol{w}_j\|_2^2 \\ &+ \frac{1}{4}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}\|H\boldsymbol{t}_i - H\boldsymbol{t}_j\|_2^2,\end{aligned} \tag{5}$$

where A is the adjacent matrix of network G. $\mathcal{R}(W,H)$ can be further decomposed as $\mathcal{R}(W,H) = \mathcal{R}_1(W) + \mathcal{R}_2(H)$, where $\mathcal{R}_1(W)$ is the regularization term on W, and $\mathcal{R}_2(H)$ is the regularization term on H. $\mathcal{R}_1(W)$ is formulated as

$$\begin{aligned}\mathcal{R}_1(W) &= \frac{1}{4}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}\|\boldsymbol{w}_i - \boldsymbol{w}_j\|_2^2 \\ &= \frac{1}{4}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}(\boldsymbol{w}_i - \boldsymbol{w}_j)^{\mathrm{T}}(\boldsymbol{w}_i - \boldsymbol{w}_j) \\ &= \frac{1}{2}\sum_{i=1}^{|\mathcal{V}|} D_{ii}\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{w}_i - \frac{1}{2}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{w}_j \\ &= \frac{1}{2}\sum_{i,j=1}^{|\mathcal{V}|} L_{ij}\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{w}_j,\end{aligned} \tag{6}$$

where $D_{ii} = \sum_{j=1}^{|\mathcal{V}|} A_{ij}, D_{ij} = 0$, for $i \neq j$ and $L = D - A$. $\mathcal{R}_2(H)$ is formulated as

$$\begin{aligned}\mathcal{R}_2(H) &= \frac{1}{4}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}\|H\boldsymbol{t}_i - H\boldsymbol{t}_j\|_2^2 \\ &= \frac{1}{4}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}\|H(\boldsymbol{t}_i - \boldsymbol{t}_j)\|_2^2 \\ &= \frac{1}{4}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}(\boldsymbol{t}_i - \boldsymbol{t}_j)^{\mathrm{T}}H^{\mathrm{T}}H(\boldsymbol{t}_i - \boldsymbol{t}_j) \\ &= \frac{1}{2}\sum_{i=1}^{|\mathcal{V}|} D_{ii}\boldsymbol{t}_i^{\mathrm{T}}H^{\mathrm{T}}H\boldsymbol{t}_i - \frac{1}{2}\sum_{i,j=1}^{|\mathcal{V}|} A_{ij}\boldsymbol{t}_i^{\mathrm{T}}H^{\mathrm{T}}H\boldsymbol{t}_j \\ &= \frac{1}{2}\sum_{i,j=1}^{|\mathcal{V}|} L_{ij}\boldsymbol{t}_i^{\mathrm{T}}H^{\mathrm{T}}H\boldsymbol{t}_j.\end{aligned} \tag{7}$$

Here, we define the new object function as

$$\begin{aligned}\mathcal{F}(W,H) = &\frac{1}{2}\|M - W^{\mathrm{T}}HT\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2) \\ &+ \mu(\mathcal{R}_1(W) + \mathcal{R}_2(H)).\end{aligned} \tag{8}$$

Then, a new optimization problem is formulated as

$$\min_{W,H} \mathcal{F}(W,H). \tag{9}$$

B. Solving the Problem

We now discuss how to derive the solution to the optimization problem (9). Though $\mathcal{F}(W,H)$ is not convex with respect to W and H, it can be easily proved that when one of W and H is fixed, $\mathcal{F}(W,H)$ is convex with respect to the other variable. After W and H are initialized properly, we can solve problem (9) with the following alternative update procedure:

$$W^{(\tau)} = \arg\min_{W} \mathcal{F}(W, H^{(\tau-1)}), \tag{10}$$

$$H^{(\tau)} = \arg\min_{H} \mathcal{F}(W^{(\tau)}, H). \tag{11}$$

To optimize W with a fixed H, we define $\boldsymbol{\omega} = \boldsymbol{vec}(W^{\mathrm{T}})$ and $\tilde{\boldsymbol{x}}_{ij} = (H\boldsymbol{t}_j) \otimes \boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is the i-th column of the $|\mathcal{V}| \times |\mathcal{V}|$ identity matrix. Then W can be updated by solving the following problem:

$$\arg\min_{\boldsymbol{\omega} \in \mathbb{R}^{|\mathcal{V}|k}} f(\boldsymbol{\omega}), \tag{12}$$

where

$$f(\boldsymbol{\omega}) = \frac{1}{2} \sum_{i,j=1}^{|\mathcal{V}|} (M_{ij} - \boldsymbol{\omega}^{\mathrm{T}} \tilde{\boldsymbol{x}}_{ij})^2 + \frac{\mu}{2} \boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{\Lambda} \boldsymbol{\omega} + \frac{\lambda}{2} \|\boldsymbol{\omega}\|_2^2, \tag{13}$$

where $\boldsymbol{\Lambda} = \mathbf{I}_k \otimes L$ with $\mathbf{I}_k$ being the $k \times k$ identity matrix.

To optimize H with W fixed, we define $\boldsymbol{\eta} = \boldsymbol{vec}(H)$ and $\tilde{\boldsymbol{y}}_{ij} = \boldsymbol{t}_j \otimes \boldsymbol{w}_i$. Then H can be updated by solving the following problem:

$$\arg\min_{\boldsymbol{\eta} \in \mathbb{R}^{kf_t}} g(\boldsymbol{\eta}), \tag{14}$$

where

$$g(\boldsymbol{\eta}) = \frac{1}{2} \sum_{i,j=1}^{|\mathcal{V}|} (M_{ij} - \boldsymbol{\eta}^{\mathrm{T}} \tilde{\boldsymbol{y}}_{ij})^2 + \frac{\mu}{2} \boldsymbol{\eta}^{\mathrm{T}} \boldsymbol{\Omega} \boldsymbol{\eta} + \frac{\lambda}{2} \|\boldsymbol{\eta}\|_2^2, \tag{15}$$

where $\boldsymbol{\Omega} = (TLT^{\mathrm{T}}) \otimes \mathbf{I}_k$.

To solve the optimization problem (12) and (14), we utilize Conjugate Gradient (CG) algorithm. The gradient of $f(\boldsymbol{\omega})$ and $g(\boldsymbol{\eta})$, and the multiplication of the Hessian matrix of $f(\boldsymbol{\omega})$ and $g(\boldsymbol{\eta})$ with vector $\boldsymbol{s}$ are calculated as follows:

$$\nabla f(\boldsymbol{\omega}) = \sum_{i,j=1}^{|\mathcal{V}|} (\boldsymbol{\omega}^{\mathrm{T}} \tilde{\boldsymbol{x}}_{ij} - M_{ij}) \tilde{\boldsymbol{x}}_{ij} + \mu \Lambda \boldsymbol{\omega} + \lambda \boldsymbol{\omega}, \tag{16}$$

$$\nabla g(\boldsymbol{\eta}) = \sum_{i,j=1}^{|\mathcal{V}|} (\boldsymbol{\eta}^{\mathrm{T}} \tilde{\boldsymbol{y}}_{ij} - M_{ij}) \tilde{\boldsymbol{y}}_{ij} + \mu \Omega \boldsymbol{\eta} + \lambda \boldsymbol{\eta}, \tag{17}$$

$$\nabla^2 f(\boldsymbol{\omega}) \boldsymbol{s} = \sum_{i,j=1}^{|\mathcal{V}|} \tilde{\boldsymbol{x}}_{ij} \tilde{\boldsymbol{x}}_{ij}^{\mathrm{T}} \boldsymbol{s} + \mu \Lambda \boldsymbol{s} + \lambda \boldsymbol{s}, \tag{18}$$

$$\nabla^2 g(\boldsymbol{\eta}) \boldsymbol{s} = \sum_{i,j=1}^{|\mathcal{V}|} \tilde{\boldsymbol{y}}_{ij} \tilde{\boldsymbol{y}}_{ij}^{\mathrm{T}} \boldsymbol{s} + \mu \Omega \boldsymbol{s} + \lambda \boldsymbol{s}. \tag{19}$$

In this way, the calculation of $\nabla f(\boldsymbol{\omega})$ and $\nabla^2 f(\boldsymbol{\omega}) \boldsymbol{s}$ needs at least $O(|\mathcal{V}|^2 k)$ time, and the calculation of $\nabla g(\boldsymbol{\eta})$ and $\nabla^2 g(\boldsymbol{\eta}) \boldsymbol{s}$ needs at least $O(|\mathcal{V}|^2 k f_t)$ time. Below, we give more efficient solutions to calculating these four variables, with the detailed derivations given in the Appendix.

1) Calculation for $\nabla f(\boldsymbol{\omega})$: As shown in the Appendix, we have proven that

$$\begin{aligned} \nabla f(\boldsymbol{w}) = \boldsymbol{vec}(W^{\mathrm{T}} HTT^{\mathrm{T}} H^{\mathrm{T}} - MT^{\mathrm{T}} H^{\mathrm{T}}) \\ + \mu \boldsymbol{vec}(LW^{\mathrm{T}}) + \lambda \boldsymbol{\omega}. \end{aligned} \tag{20}$$

By changing the order of matrix multiplication in (20) carefully as

$$\begin{aligned} \nabla f(\boldsymbol{w}) = \boldsymbol{vec}(W^{\mathrm{T}}((HT)(HT)^{\mathrm{T}}) - M(HT)^{\mathrm{T}}) \\ + \mu \boldsymbol{vec}(LW^{\mathrm{T}}) + \lambda \boldsymbol{\omega}, \end{aligned} \tag{21}$$

the calculation of $\nabla f(\boldsymbol{w})$ requires only $O(|\mathcal{V}|k^2 + |\mathcal{V}|kf_t + nnz(L)k + nnz(M)k)$ time. For most real-world networks, edges are sparse, i.e., $O(|\mathcal{E}|) = O(|\mathcal{V}|)$, which indicates that $nnz(L), nnz(M) \ll |\mathcal{V}|^2$. Thus, calculating $\nabla f(\boldsymbol{w})$ with (20) is more efficient.

2) Calculation for $\nabla^2 f(\boldsymbol{\omega}) \boldsymbol{s}$: In Appendix, we have shown that

$$\nabla^2 f(\boldsymbol{\omega}) \boldsymbol{s} = \boldsymbol{vec}(SHTT^{\mathrm{T}} H^{\mathrm{T}}) + \mu \boldsymbol{vec}(LS) + \lambda \boldsymbol{s}, \tag{22}$$

where $S \in \mathbb{R}^{|\mathcal{V}| \times k}$ satisfies $\boldsymbol{s} = \boldsymbol{vec}(S)$. Similarly, in implementation, $f(\boldsymbol{\omega}) \boldsymbol{s}$ is calculated as

$$\nabla^2 f(\boldsymbol{\omega}) \boldsymbol{s} = \boldsymbol{vec}(S((HT)(HT)^{\mathrm{T}})) + \mu \boldsymbol{vec}(LS) + \lambda \boldsymbol{s}. \tag{23}$$

The above calculation of $f(\boldsymbol{\omega}) \boldsymbol{s}$ requires only $O(|\mathcal{V}|k^2 + |\mathcal{V}|kf_t + nnz(L)k)$ time.

3) Calculation for $\nabla g(\boldsymbol{\eta})$: As is proved in Appendix,

$$\begin{aligned} \nabla g(\boldsymbol{\eta}) = \boldsymbol{vec}(WW^{\mathrm{T}} HTT^{\mathrm{T}} - WMT^{\mathrm{T}}) \\ + \mu \boldsymbol{vec}(H(TLT^{\mathrm{T}})^{\mathrm{T}}) + \lambda \boldsymbol{\eta}. \end{aligned} \tag{24}$$

For implementation efficiency, $\nabla g(\boldsymbol{\eta})$ is calculated as

$$\begin{aligned} \nabla g(\boldsymbol{\eta}) = \boldsymbol{vec}(W((W^{\mathrm{T}} H)(TT^{\mathrm{T}})) - W(MT^{\mathrm{T}})) \\ + \mu \boldsymbol{vec}(H(TLT^{\mathrm{T}})^{\mathrm{T}}) + \lambda \boldsymbol{\eta}, \end{aligned} \tag{25}$$

which requires only $O(|\mathcal{V}|kf_t + |\mathcal{V}|f_t^2 + nnz(L)f_t + nnz(M)f_t)$ time.

4) Calculation for $\nabla^2 g(\boldsymbol{\eta}) \boldsymbol{s}$: It can also be proved that

$$\begin{aligned} \nabla^2 g(\boldsymbol{\eta}) \boldsymbol{s} = \boldsymbol{vec}(WW^{\mathrm{T}} STT^{\mathrm{T}}) \\ + \mu \boldsymbol{vec}(S(TLT^{\mathrm{T}})^{\mathrm{T}}) + \lambda \boldsymbol{s}. \end{aligned} \tag{26}$$

For implementation efficiency, $\nabla^2 g(\boldsymbol{\eta}) \boldsymbol{s}$ is calculated as

$$\begin{aligned} \nabla^2 g(\boldsymbol{\eta}) \boldsymbol{s} = \boldsymbol{vec}(W((W^{\mathrm{T}} S)(TT^{\mathrm{T}}))) \\ + \mu \boldsymbol{vec}(S(TLT^{\mathrm{T}})^{\mathrm{T}}) + \lambda \boldsymbol{s}, \end{aligned} \tag{27}$$

which needs only $O(|\mathcal{V}|kf_t + |\mathcal{V}|f_t^2 + nnz(L)f_t)$ time.

V. Experiments

To evaluate the quality of learned representations, we conduct multi-class node classification on real-world networks.

A. Benchmark Networks

Four information networks, Cora, Citeseer, Wikipedia, and PubMed[1] are used in our experiments. A summary of the four networks is given in Table II. For each network, link directions and node's self-links are not considered. Two nodes have a link, if either of them is directly linked to the other. A detailed introduction to the four networks is given as follows:

Cora: It contains 2,708 machine learning publications from seven classes, and 5,249 citation links among these papers. Each paper is represented by a 0/1 vector of 1,433 dimensions indicating the presence/absence of the corresponding words.

Citeseer: It contains 3,312 scientific publications from six classes. Among these papers, there are 4,732 links. Each paper is represented by a 0/1 vector of 3,703 dimensions.

[1]http://linqs.cs.umd.edu/projects/projects/lbc/index.html

TABLE II
SUMMARY OF FOUR REAL-WORLD NETWORKS

Data Set	Cora	Citeseer	PubMed	Wikipedia
# of Nodes	2,708	3,312	19,717	2,405
# of Links	5,429	4,732	44,338	17,981
# of Features	1,433	3,703	500	4,973
# of Class	7	6	3	19

PubMed: It contains 19,717 scientific publications related to diabetes in three classes. Each paper is represented by a TF-IDF weighted word vector from a dictionary composed by 500 unique words. There are 44,338 links among them.

Wikipedia: It contains 2,405 Wikipedia articles from 19 classes. Each article is represented by a 4,973-dimensional TF-IDF vector. There are 17,981 hyperlinks among these articles.

B. Experimental Settings

In our experiments, we focus on multi-class node classification in sparsely labeled networks, in which we vary the training ratio from 1% to 10% by an increment of 1%. Because node classification with a small amount of training data is a difficult task, the effectiveness of different node representations can be clearly compared under this setting. For each training ratio, we randomly select a small number of labeled nodes for training, and the rest of unlabeled nodes are held out for testing. We repeat this process ten times and report the averaged Micro-F_1 and Macro-F_1 values as our evaluation criteria. Let TP_i, FP_i and FN_i denote true positive, false positive and false negative for class i, Micro-F_1 and Macro-F_1 for m-class node classification are defined as follows.

$$\text{Micro-}F_1 = \frac{2\sum_{i=1}^{m} TP_i}{2\sum_{i=1}^{m} TP_i + \sum_{i=1}^{m} FP_i + \sum_{i=1}^{m} FN_i}, \quad (28)$$

$$\text{Macro-}F_1 = \frac{1}{m}\sum_{i=1}^{m} \frac{2TP_i}{2TP_i + FP_i + FN_i}. \quad (29)$$

For all NRL algorithms, we use the linear SVM implemented by Liblinear [21] to perform multi-class classification. We reduce the dimension of node text features to 200 by applying Singular Value Decomposition (SVD) on the TF-IDF matrix. For our proposed HSCA algorithm, k is set to 120, λ is set to 0.2, μ is set to 0.1 for Cora, Citeseer and PubMed, and for Wikipedia, μ is set to 0.04. W and H are initialized by random numbers and the iterative update for W and H stops when $\|W^{(\tau)} - W^{(\tau-1)}\|_F^2 + \|H^{(\tau)} - H^{(\tau-1)}\|_F^2 \leq 0.01$.

C. Baseline Methods

Our proposed HSCA algorithm is compared against the following baseline methods:

Text: This baseline simply uses 200-dimensional text features as node representations, which exploit only node content.

DeepWalk [1]: DeepWalk is a popularly studied NRL algorithm that utilizes structural context for learning network representations. Default parameters are set for this algorithm.

DeepWalk+Text: The representations learned by DeepWalk and text features are simply concatenated together to serve as new node representations, which consider structural context information and node content information.

LINE [2]: LINE is a recently proposed NRL algorithm that considers the first-order proximity and second-order proximity. Two types of representations are separately learned and concatenated together to form the final representation. Default parameters are also used.

LINE+Text: This baseline concatenates LINE's representations and node text features to form new node representations.

GraRep [10]: GraRep is one state-of-the-art NRL algorithm that learns node representations by exploiting more global structural context than LINE. Parameters for this algorithm are also set as default values.

GraRep+Text: This method combines the representations learned by GraRep and node content features to generate new node representations.

TADW [11]: TADW utilizes both structural context and node content information to learn node representations, which is shown to outperform the naive combination of DeepWalk's representations and node content text features. For fair comparison, the parameter k is set to 120 as done in our proposed algorithm, and other parameters are set as default values.

D. Comparison of Network Representations

The Micro-F_1 and Macro-F_1 values for multi-class classification with different network representations on different networks are reported in Table III-VI, respectively. For each training ratio, the largest Micro-F_1 and Macro-F_1 values are **bold-faced**. As the PubMed network is too large for GraRep to run, Micro-F_1 and Macro-F_1 values for GraRep and GraRep+Text are not available in Table V.

From the four tables, it is clear that HSCA outperforms all other baselines in terms of both Micro-F_1 and Macro-F_1 in most cases, with an exception of Macro-F_1 on Wikipedia with 8% and 9% training ratios (Table VI), where HSCA performs slightly worse than GraRep+Text. TADW is the second best performer because it captures the interactions between structural context and node content. However, as TADW does not consider homophily, its representations are less effective than those learned by HSCA. This suggests that homophily is one important property to preserve, especially when node content is exploited for representation learning.

By comparing HSCA with other baselines except LINE+text and GraRep+text, we can infer that with more information sources imported, the representations learned by HSCA can achieve better classification performance. The comparison between HSCA, LINE+Text, and GraRep+Text indicates that, although they all utilize three information sources (homophily, contextual structure, and node content), LINE+Text and GraRep+Text just naively combine different information sources, while HSCA better captures their interplay by embedding the network into a single latent representation space, and further enforce the homophily constraints in the new space. As a result, it leads to more informative network representations.

TABLE III
CLASSIFICATION RESULTS ON CORA

	Training Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-$F_1(\%)$	Text	28.59	34.18	38.37	42.77	46.33	49.32	51.78	52.88	56.75	57.03
	DeepWalk	46.18	58.49	62.75	67.00	70.32	71.24	72.49	73.13	73.51	74.43
	DeepWalk+Text	44.03	56.67	61.66	67.62	71.22	72.57	74.07	74.48	75.85	76.85
	LINE	31.45	37.17	40.34	43.15	45.44	46.81	47.52	47.58	49.72	50.11
	LINE+Text	32.49	39.09	43.71	48.19	51.67	53.84	55.71	56.87	59.37	60.45
	GraRep	40.70	51.80	58.79	62.20	67.85	69.61	71.24	72.04	73.40	73.91
	GraRep+Text	40.25	51.78	58.88	63.66	69.34	71.13	72.95	73.50	75.67	76.16
	TADW	43.00	56.78	64.10	70.21	75.54	76.73	78.93	79.79	80.95	82.06
	HSCA	**49.09**	**64.85**	**69.79**	**75.27**	**79.10**	**79.80**	**81.27**	**81.99**	**82.53**	**83.69**
Macro-$F_1(\%)$	Text	14.21	20.18	26.52	32.17	38.57	41.38	45.70	47.29	51.95	52.59
	DeepWalk	33.22	53.49	56.85	62.32	67.79	68.00	70.22	71.10	71.55	72.94
	DeepWalk+Text	30.38	50.93	55.53	62.85	68.51	69.23	72.09	72.52	73.92	75.48
	LINE	20.51	29.26	33.68	37.44	42.81	43.65	45.15	45.28	47.46	48.17
	LINE+Text	20.30	29.67	35.68	41.81	47.99	49.72	52.69	54.11	57.04	58.41
	GraRep	32.64	48.37	55.57	59.21	66.36	67.59	69.80	70.76	72.19	72.62
	GraRep+Text	30.14	47.42	54.83	60.26	67.83	69.04	71.37	72.16	74.27	74.97
	TADW	30.48	51.01	58.38	65.98	73.64	74.27	77.25	78.10	79.28	80.82
	HSCA	**37.31**	**60.70**	**65.38**	**71.94**	**77.48**	**77.89**	**79.92**	**80.53**	**81.05**	**82.45**

TABLE IV
CLASSIFICATION RESULTS ON CITESEER

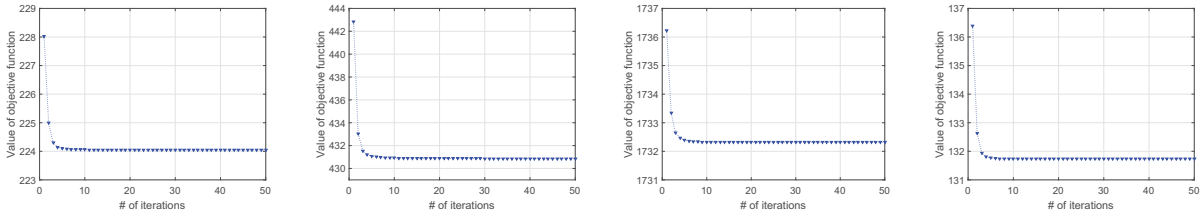
	Training Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-$F_1(\%)$	Text	27.09	35.26	41.09	43.65	46.11	49.75	51.91	55.37	56.53	58.02
	DeepWalk	30.71	38.60	42.49	43.29	44.05	45.75	45.92	46.87	47.96	47.72
	DeepWalk+Text	32.51	42.35	48.02	49.61	52.07	54.10	55.53	57.98	58.99	59.71
	LINE	22.36	26.09	29.08	30.19	31.05	33.03	33.65	34.30	35.35	36.24
	LINE+Text	26.55	33.28	38.35	40.20	42.12	45.74	47.70	50.38	51.28	52.70
	GraRep	19.34	19.65	20.04	20.34	19.79	19.98	19.49	19.99	19.69	20.41
	GraRep+Text	27.09	35.23	41.14	43.66	46.13	49.75	51.86	55.41	56.54	58.09
	TADW	36.10	49.39	56.97	60.14	63.11	64.17	65.97	68.01	68.74	68.99
	HSCA	**37.66**	**51.46**	**59.03**	**61.90**	**64.33**	**65.59**	**67.03**	**68.78**	**69.42**	**69.63**
Macro-$F_1(\%)$	Text	19.02	29.34	36.05	38.10	40.94	44.11	46.77	50.16	51.56	52.73
	DeepWalk	23.09	32.12	36.22	36.46	37.50	38.87	39.06	39.63	41.16	40.79
	DeepWalk+Text	24.74	36.51	42.24	43.56	46.61	48.04	49.94	52.45	53.80	54.36
	LINE	16.00	21.71	25.70	26.53	27.73	29.35	30.29	31.06	32.26	32.93
	LINE+Text	19.83	28.33	34.08	35.53	38.06	41.07	43.50	46.12	47.21	48.37
	GraRep	5.40	5.47	5.56	5.63	5.51	5.55	5.43	5.55	5.48	5.65
	GraRep+Text	19.01	29.33	36.08	38.09	40.94	44.11	46.70	50.20	51.57	52.80
	TADW	28.67	43.60	51.46	53.94	57.24	57.83	60.20	62.28	63.44	63.61
	HSCA	**30.04**	**45.66**	**53.28**	**55.65**	**58.39**	**59.08**	**61.21**	**62.91**	**64.20**	**64.03**

TABLE V
CLASSIFICATION RESULTS ON PUBMED

	Training Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-$F_1(\%)$	Text	52.81	62.16	67.43	71.69	74.53	76.29	77.99	78.55	79.60	80.24
	DeepWalk	67.98	73.82	75.90	76.80	77.23	77.38	77.93	77.82	78.19	78.23
	DeepWalk+Text	71.20	78.07	80.27	81.59	82.32	82.71	83.21	83.32	83.72	83.92
	LINE	52.01	56.88	60.54	62.32	63.51	64.46	65.48	65.71	66.67	66.93
	LINE+Text	59.29	67.90	72.75	75.44	77.33	78.57	79.60	80.16	81.25	81.57
	GraRep	-	-	-	-	-	-	-	-	-	-
	GraRep+Text	-	-	-	-	-	-	-	-	-	-
	TADW	69.71	78.93	81.48	82.75	83.35	83.74	84.16	84.21	84.33	84.59
	HSCA	**75.31**	**81.55**	**82.93**	**83.69**	**84.10**	**84.29**	**84.73**	**84.73**	**84.79**	**85.05**
Macro-$F_1(\%)$	Text	39.75	53.10	62.58	68.56	72.46	74.72	77.07	77.66	79.09	79.86
	DeepWalk	55.80	68.10	72.35	73.67	74.45	74.65	75.50	75.25	75.87	75.89
	DeepWalk+Text	62.70	74.87	78.49	80.10	81.12	81.56	82.21	82.29	82.87	83.04
	LINE	39.39	46.68	52.91	55.85	58.13	59.59	61.31	61.32	63.06	63.44
	LINE+Text	49.25	62.52	70.01	73.56	76.03	77.51	78.78	79.40	80.74	81.11
	GraRep	-	-	-	-	-	-	-	-	-	-
	GraRep+Text	-	-	-	-	-	-	-	-	-	-
	TADW	62.45	76.99	80.68	82.07	82.87	83.28	83.79	83.81	83.98	84.25
	HSCA	**71.39**	**80.61**	**82.45**	**83.24**	**83.77**	**83.94**	**84.43**	**84.41**	**84.51**	**84.75**

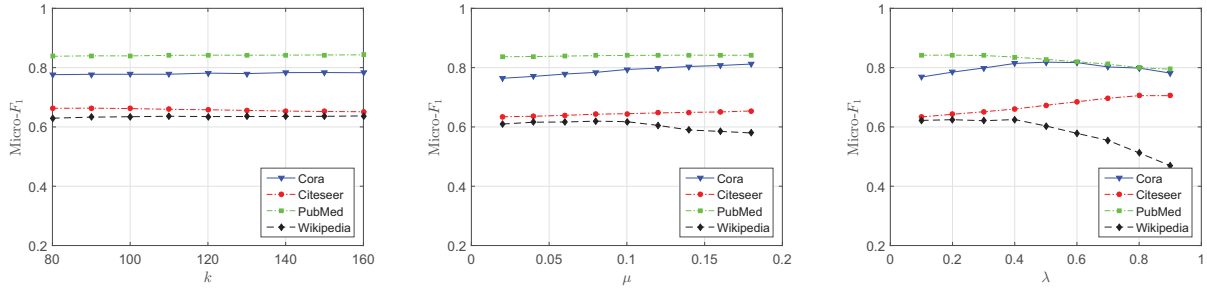
TABLE VI
CLASSIFICATION RESULTS ON WIKIPEDIA

	Training Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F_1(%)	Text	29.80	41.79	46.90	51.75	55.86	60.44	61.14	64.67	65.80	66.38
	DeepWalk	36.07	44.32	47.92	51.62	52.78	54.74	55.08	56.65	58.00	58.43
	DeepWalk+Text	37.41	48.36	52.36	56.35	59.08	62.65	62.99	65.38	66.99	67.28
	LINE	25.11	30.92	34.57	38.67	39.88	43.01	43.77	46.65	47.49	49.28
	LINE+Text	27.84	36.01	40.42	45.50	48.21	52.50	53.36	56.66	58.87	60.09
	GraRep	14.85	15.40	14.41	15.43	16.16	16.46	15.97	15.62	15.62	16.77
	GraRep+Text	29.75	41.81	46.86	51.66	55.95	60.46	61.18	64.67	65.85	66.33
	TADW	36.72	50.10	52.60	58.25	60.18	63.63	64.22	65.44	67.00	66.89
	HSCA	**39.92**	**53.85**	**55.78**	**60.29**	**62.70**	**65.49**	**65.45**	**67.04**	**68.44**	**68.79**
Macro-F_1(%)	Text	14.98	26.96	30.09	34.43	39.87	41.51	43.34	47.60	**48.50**	48.14
	DeepWalk	19.09	28.33	28.85	32.55	34.16	35.81	36.42	37.26	38.47	39.17
	DeepWalk+Text	19.99	31.25	32.95	36.83	40.52	42.36	43.35	46.41	47.54	47.73
	LINE	13.46	20.85	22.70	26.57	27.33	29.98	30.71	32.18	33.21	34.90
	LINE+Text	14.67	23.97	26.34	30.87	33.82	36.60	38.15	40.78	42.44	43.28
	GraRep	1.35	1.40	1.32	1.40	1.46	1.49	1.45	1.42	1.42	1.51
	GraRep+Text	14.97	26.91	30.04	34.35	39.92	41.56	43.28	**47.63**	**48.49**	48.11
	TADW	19.83	32.95	33.56	38.11	41.25	42.69	44.37	45.59	46.92	46.55
	HSCA	**22.07**	**35.05**	**35.41**	**39.22**	**43.72**	**44.52**	**45.03**	46.73	48.37	**48.44**



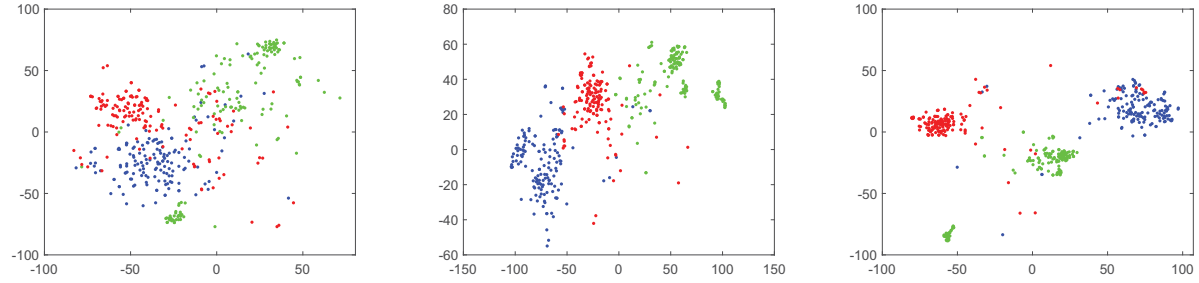
(a) Cora (b) Citeseer (c) PubMed (d) Wikipedia

Fig. 2. Convergence of the objective function



(a) k (b) μ (c) λ

Fig. 3. Micro-F_1 values with respect to different values of k, μ and λ



(a) DeepWalk (b) TADW (c) HSCA

Fig. 4. Visualization of network representations learned by different algorithms

For DeepWalk, LINE and GraRep, when naively combined with text features, their learned representations can achieve better performance. This proves that node content is helpful for learning informative node representations. On Cora, GraRep is observed to remarkably outperform LINE, which demonstrates the usefulness of more global structural context. However, GraRep does not perform well on Citeseer and Wikipedia. This is possibly because, by naively combing representations learned from different k-step proximity, GraRep may "wash out" the effectiveness of the most important component, leading to unsatisfactory performance.

E. Convergence Analysis

We also conduct experiments to examine the convergence of the solution to problem (9). The values of the objective function (8) from iteration 1 to iteration 50 on all four networks are reported in Fig. 2. Fig. 2 shows that the algorithm can quickly convergence within only 5 iterations on all networks.

F. Parameter Sensitivity

In our algorithm, there are three parameters, k, μ and λ. We fix any two parameters and investigate the sensitivity of the third one in turns. Fig. 3(a)-3(c) show the averaged Micro-F_1 values on all four networks by varying k, μ and λ, respectively. Here, we set the training ratio as 5%. From Fig. 3(a) and 3(b), we can see that, HSCA is relatively robust to the change of parameters k and μ. As the value of λ increases, the change of Micro-F_1 follows different trends on different networks. Yet, overall, the best performance can be achieved when λ is set to be 0.4 or 0.5.

G. Visualization of Learned Representations

To validate that the representations learned by our HSCA algorithm are indeed more informative than those learned by TADW and DeepWalk, we project the representations learned by the three algorithms into 2-dimensional space using the t-SNE package [22]. We randomly select 150 nodes from three subcategories of Cora, including Genetic Algorithms, Rule Learning and Reinforcement Learning, and plot the visualization results in Fig. 4. By comparing Fig. 4(a) and Fig. 4(b), we can observe that different classes of data points in Fig. 4(b) are better separated from each other than those in Fig. 4(a). This indicates that with imported node content, TADW's representations are more informative than those learned by DeepWalk. Compared to Fig. 4(b), data points in Fig. 4(c) tend to have a cluster structure; data points belonging to the same class are closer to each other, while the ones falling into different classes are far apart separated. This observation confirms that, by preserving homophily, HSCA's representations have more discriminative power for node classification.

VI. Conclusion

In this paper, we proposed a new NRL algorithm called HSCA that effectively augments information from (1) network homophily, (2) structural context, and (3) node content into a new optimization objective function for learning network representations. An efficient algorithm is also derived to solve the objective function. A unique feature of our HSCA algorithm is that, it effectively embeds a network into a single latent representation space that captures the interplay between the three information sources. As a result, the learned representations not only preserve network structure and node content information, but also explicitly follow the social homophily constraints in the new space. We validated the effectiveness of the HSCA algorithm on real-world networks. Experiments and visualization results demonstrate that the node representations learned by HSCA can be more effectively used as features for tasks such as multi-class node classification, compared to those learned by the state-of-the-art NRL algorithms.

Acknowledgment

This work is partially supported by the Australian Research Council (ARC) under discovery grant DP140100545, and by the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning. Daokun Zhang is supported by China Scholarship Council (CSC) with No. 201506300082 and a supplementary postgraduate scholarship from CSIRO.

References

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of SIGKDD*. ACM, 2014, pp. 701–710.

[2] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proceedings of WWW*, 2015, pp. 1067–1077.

[3] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[4] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 303–336, 2014.

[5] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[6] M. McPherson, L. Smith-Lovin, and J. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, pp. 415–444, 2001.

[7] L. M. Aiello, A. Barrat, R. Schifanella, C. Cattuto, B. Markines, and F. Menczer, "Friendship prediction and homophily in social media," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 2, 2012.

[8] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proceedings of SIGKDD*. ACM, 2009, pp. 817–826.

[9] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of NIPS*, 2013, pp. 3111–3119.

[10] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proceedings of CIKM*. ACM, 2015, pp. 891–900.

[11] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proceedings of IJCAI*, 2015, pp. 2111–2117.
[12] T. F. Cox and M. A. Cox, *Multidimensional scaling*. CRC press, 2000.
[13] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[14] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[15] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering." in *In Proceedings of NIPS*, vol. 14, 2001, pp. 585–591.
[16] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 447–478, 2011.
[17] ——, "Scalable learning of collective behavior based on sparse social dimensions," in *Proceedings of CIKM*. ACM, 2009, pp. 1107–1116.
[18] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.
[19] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, p. 036104, 2006.
[20] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model." in *Aistats*, vol. 5, 2005, pp. 246–252.
[21] 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*, vol. 9, pp. 1871–1874, 2008.
[22] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

APPENDIX

A. Calculation for $\nabla f(\boldsymbol{\omega})$

As $\tilde{\boldsymbol{x}}_{ij} = (H\boldsymbol{t}_j) \otimes \boldsymbol{e}_i = \boldsymbol{vec}(\boldsymbol{e}_i \boldsymbol{t}_j^{\mathrm{T}} H^{\mathrm{T}})$, we calculate

$$\sum_{i,j=1}^{|\mathcal{V}|} (\boldsymbol{\omega}^{\mathrm{T}} \tilde{\boldsymbol{x}}_{ij} - M_{ij}) \boldsymbol{e}_i \boldsymbol{t}_j^{\mathrm{T}} H^{\mathrm{T}} = F T^{\mathrm{T}} H^{\mathrm{T}}, \tag{30}$$

where $F_{ij} = \boldsymbol{\omega}^{\mathrm{T}} \tilde{\boldsymbol{x}}_{ij} - M_{ij}$, i.e., $F = W^{\mathrm{T}} H T - M$. Therefore,

$$\sum_{i,j=1}^{|\mathcal{V}|} (\boldsymbol{\omega}^{\mathrm{T}} \tilde{\boldsymbol{x}}_{ij} - M_{ij}) \boldsymbol{e}_i \boldsymbol{t}_j^{\mathrm{T}} H^{\mathrm{T}} = W^{\mathrm{T}} H T T^{\mathrm{T}} H^{\mathrm{T}} - M T^{\mathrm{T}} H^{\mathrm{T}}. \tag{31}$$

Using the identity $(B^{\mathrm{T}} \otimes A)\boldsymbol{vec}(X) = \boldsymbol{vec}(AXB)$, $\Lambda\boldsymbol{\omega}$ can be expanded as

$$\Lambda\boldsymbol{\omega} = (\mathrm{I}_k \otimes L)\boldsymbol{vec}(W^{\mathrm{T}}) = \boldsymbol{vec}(LW^{\mathrm{T}}). \tag{32}$$

Thus,

$$\begin{aligned} \nabla f(\boldsymbol{w}) = {} & \boldsymbol{vec}(W^{\mathrm{T}} H T T^{\mathrm{T}} H^{\mathrm{T}} - M T^{\mathrm{T}} H^{\mathrm{T}}) \\ & + \mu \boldsymbol{vec}(LW^{\mathrm{T}}) + \lambda\boldsymbol{\omega}. \end{aligned} \tag{33}$$

B. Calculation for $\nabla^2 f(\boldsymbol{\omega})\boldsymbol{s}$

After substituting $\tilde{\boldsymbol{x}}_{ij} = (H\boldsymbol{t}_j) \otimes \boldsymbol{e}_i$, we have

$$\sum_{i,j=1}^{|\mathcal{V}|} \tilde{\boldsymbol{x}}_{ij} \tilde{\boldsymbol{x}}_{ij}^{\mathrm{T}} \boldsymbol{s} = \sum_{i,j=1}^{|\mathcal{V}|} \left[(H\boldsymbol{t}_j \boldsymbol{t}_j^{\mathrm{T}} H^{\mathrm{T}}) \otimes (\boldsymbol{e}_i \boldsymbol{e}_i^{\mathrm{T}}) \right] \boldsymbol{s}. \tag{34}$$

Let $S \in \mathbb{R}^{|\mathcal{V}| \times k}$ such that $\boldsymbol{s} = \boldsymbol{vec}(S)$. Similar to (32),

$$\left[(H\boldsymbol{t}_j \boldsymbol{t}_j^{\mathrm{T}} H^{\mathrm{T}}) \otimes (\boldsymbol{e}_i \boldsymbol{e}_i^{\mathrm{T}}) \right] \boldsymbol{s} = \boldsymbol{vec}(\boldsymbol{e}_i \boldsymbol{e}_i^{\mathrm{T}} S H \boldsymbol{t}_j \boldsymbol{t}_j^{\mathrm{T}} H^{\mathrm{T}}). \tag{35}$$

We can calculate

$$\begin{aligned} \sum_{i,j=1}^{|\mathcal{V}|} \boldsymbol{e}_i \boldsymbol{e}_i^{\mathrm{T}} S H \boldsymbol{t}_j \boldsymbol{t}_j^{\mathrm{T}} H^{\mathrm{T}} &= \sum_{i=1}^{|\mathcal{V}|} \boldsymbol{e}_i \left(\sum_{j=1}^{|\mathcal{V}|} \boldsymbol{e}_i^{\mathrm{T}} S H \boldsymbol{t}_j \boldsymbol{t}_j^{\mathrm{T}} H^{\mathrm{T}} \right) \\ &= \sum_{i=1}^{|\mathcal{V}|} \boldsymbol{e}_i \left(\sum_{j=1}^{|\mathcal{V}|} U_{ij} \boldsymbol{t}_j^{\mathrm{T}} H^{\mathrm{T}} \right) \\ &= U T^{\mathrm{T}} H^{\mathrm{T}}, \end{aligned} \tag{36}$$

where $U_{ij} = \boldsymbol{e}_i^{\mathrm{T}} S H \boldsymbol{t}_j$, i.e., $U = SHT$. Similar to (32),

$$\Lambda\boldsymbol{s} = (\mathrm{I}_k \otimes L)\boldsymbol{vec}(S) = \boldsymbol{vec}(LS). \tag{37}$$

Hence, (18) can be finally rewritten as

$$\begin{aligned} \nabla^2 f(\boldsymbol{\omega})\boldsymbol{s} &= \boldsymbol{vec}(UT^{\mathrm{T}} H^{\mathrm{T}}) + \mu \boldsymbol{vec}(LS) + \lambda\boldsymbol{s} \\ &= \boldsymbol{vec}(SHTT^{\mathrm{T}} H^{\mathrm{T}}) + \mu \boldsymbol{vec}(LS) + \lambda\boldsymbol{s}. \end{aligned} \tag{38}$$

C. Calculation for $\nabla g(\boldsymbol{\eta})$

As $\tilde{\boldsymbol{y}}_{ij} = \boldsymbol{t}_j \otimes \boldsymbol{w}_i = \boldsymbol{vec}(\boldsymbol{w}_i \boldsymbol{t}_j^{\mathrm{T}})$, we calculate

$$\sum_{i,j=1}^{|\mathcal{V}|} (\boldsymbol{\eta}^{\mathrm{T}} \tilde{\boldsymbol{y}}_{ij} - M_{ij}) \boldsymbol{w}_i \boldsymbol{t}_j^{\mathrm{T}} = W F T^{\mathrm{T}} = W W^{\mathrm{T}} H T T^{T} - W M T^{T}. \tag{39}$$

where $F_{ij} = \boldsymbol{\eta}^{\mathrm{T}} \tilde{\boldsymbol{y}}_{ij} - M_{ij}$ i.e., $F = W^{\mathrm{T}} H T - M$. Then,

$$\Omega\boldsymbol{\eta} = ((TLT^{\mathrm{T}}) \otimes \mathrm{I}_k)\boldsymbol{vec}(H) = \boldsymbol{vec}(H(TLT^{\mathrm{T}})^{\mathrm{T}}). \tag{40}$$

Thus,

$$\begin{aligned} \nabla g(\boldsymbol{\eta}) = {} & \boldsymbol{vec}(WW^{\mathrm{T}} H T T^{\mathrm{T}} - W M T^{\mathrm{T}}) \\ & + \mu \boldsymbol{vec}(H(TLT^{\mathrm{T}})^{\mathrm{T}}) + \lambda\boldsymbol{\eta}. \end{aligned} \tag{41}$$

D. Calculation for $\nabla^2 g(\boldsymbol{\eta})\boldsymbol{s}$

After substituting $\tilde{\boldsymbol{y}}_{ij} = \boldsymbol{t}_j \otimes \boldsymbol{w}_i$, we have

$$\sum_{i,j=1}^{|\mathcal{V}|} \tilde{\boldsymbol{y}}_{ij} \tilde{\boldsymbol{y}}_{ij}^{\mathrm{T}} \boldsymbol{s} = \sum_{i,j=1}^{|\mathcal{V}|} \left[(\boldsymbol{t}_j \boldsymbol{t}_j^{\mathrm{T}}) \otimes (\boldsymbol{w}_i \boldsymbol{w}_i^{\mathrm{T}}) \right] \boldsymbol{s}. \tag{42}$$

Let $S \in \mathbb{R}^{k \times f_t}$, such that $\boldsymbol{s} = \boldsymbol{vec}(S)$. Similar to (32),

$$\left[(\boldsymbol{t}_j \boldsymbol{t}_j^{\mathrm{T}}) \otimes (\boldsymbol{w}_i \boldsymbol{w}_i^{\mathrm{T}}) \right] \boldsymbol{s} = \boldsymbol{vec}(\boldsymbol{w}_i \boldsymbol{w}_i^{\mathrm{T}} S \boldsymbol{t}_j \boldsymbol{t}_j^{\mathrm{T}}). \tag{43}$$

Similar to (36),

$$\sum_{i,j=1}^{|\mathcal{V}|} \boldsymbol{w}_i \boldsymbol{w}_i^{\mathrm{T}} S \boldsymbol{t}_j \boldsymbol{t}_j^{\mathrm{T}} = W V T^{\mathrm{T}}, \tag{44}$$

where $V_{ij} = \boldsymbol{w}_i^{\mathrm{T}} S \boldsymbol{t}_j$, i.e., $V = W^{\mathrm{T}} S T$. Similar to (32),

$$\Omega\boldsymbol{s} = ((TLT^{\mathrm{T}}) \otimes \mathrm{I}_k)\boldsymbol{vec}(S) = \boldsymbol{vec}(S(TLT^{\mathrm{T}})^{\mathrm{T}}). \tag{45}$$

Hence, (19) can be rewritten as

$$\begin{aligned} \nabla^2 g(\boldsymbol{\eta})\boldsymbol{s} &= \boldsymbol{vec}(WVT^{\mathrm{T}}) + \mu \boldsymbol{vec}(S(TLT^{\mathrm{T}})^{\mathrm{T}}) + \lambda\boldsymbol{s} \\ &= \boldsymbol{vec}(WW^{\mathrm{T}} S T T^{\mathrm{T}}) + \mu \boldsymbol{vec}(S(TLT^{\mathrm{T}})^{\mathrm{T}}) + \lambda\boldsymbol{s}. \end{aligned} \tag{46}$$