User Profile Preserving Social Network Embedding

Daokun Zhang[a], **Jie Yin**[b], **Xingquan Zhu**[c,d], **Chengqi Zhang**[a]
[a] Centre for Artificial Intelligence, FEIT, University of Technology Sydney, Australia
[b] Data61, CSIRO, Australia
[c] Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, USA
[d] School of Computer Science, Fudan University, China
Daokun.Zhang@student.uts.edu.au; Jie.Yin@csiro.au; xqzhu@cse.fau.edu; Chengqi.Zhang@uts.edu.au

Abstract

This paper addresses social network embedding, which aims to embed social network nodes, including user profile information, into a latent low-dimensional space. Most of the existing works on network embedding only consider network structure, but ignore user-generated content that could be potentially helpful in learning a better joint network representation. Different from rich node content in citation networks, user profile information in social networks is useful but noisy, sparse, and incomplete. To properly utilize this information, we propose a new algorithm called User Profile Preserving Social Network Embedding (UPP-SNE), which incorporates user profile with network structure to jointly learn a vector representation of a social network. The theme of UPP-SNE is to embed user profile information via a non-linear mapping into a consistent subspace, where network structure is seamlessly encoded to jointly learn informative node representations. Extensive experiments on four real-world social networks show that compared to state-of-the-art baselines, our method learns better social network representations and achieves substantial performance gains in node classification and clustering tasks.

1 Introduction

The huge growth of online social networks, e.g., Facebook, Twitter, Google Talk, Wechat, etc., has revolutionized a new way for people to connect, express themselves, and share content with others in today's cyber society. Users in online social networks are connected with each other to form a social graph (e.g., the friendship graph). One of the most critical problems in social network analysis is the automatic classification of users into meaningful groups based on their social graphs, which has many useful practical applications such as user search, targeted advertising and recommendation systems. Therefore, it is essential to accurately learn useful information from social networks. One promising strategy is to learn a vector representation of a social network: each network node is represented as a low-dimensional vector such that the information conveyed by the original social network can be effectively captured. As a result, existing machine learning methods can be directly applied in the low-dimensional vector space to perform network analytic tasks such as node classification, network clustering, etc.

Recently, a series of algorithms have been proposed for network representation learning (NRL), such as DeepWalk [Perozzi *et al.*, 2014], LINE [Tang *et al.*, 2015], GraRep [Cao *et al.*, 2015], and node2vec [Grover and Leskovec, 2016]. These approaches have been shown to be effective in a variety of network analytic tasks, ranging from node classification [Sen *et al.*, 2008], anomaly detection [Bhuyan *et al.*, 2014], community detection [Yang *et al.*, 2013], to link prediction [Lü and Zhou, 2011]. However, most of them have considered network structure only, *e.g.*, the links between nodes, but ignored other user-generated content (*e.g.*, text, user profiles) that could potentially benefit network representation learning and subsequent analytic tasks.

In this paper, we are mainly concerned about the problem of *social network embedding*, which embeds each user in a social network into a latent low-dimensional space. In social networks, users are not only connected by social relationships (*e.g.*, friendship or the follower-followee relationship), but they are also associated with user profile information, consisting of attributes such as gender, geographic location, interests, or school/affiliation. Such profile information can reflect and affect the forming of community structures and social circles [Leskovec and Mcauley, 2012; Yang *et al.*, 2013]. Motivated by the fact that user profile information is potentially helpful in learning a better joint network representation, we focus on studying how user profile information can be leveraged and incorporated into the learning of social network representations.

Indeed, several very recently developed algorithms [Yang *et al.*, 2015; Pan *et al.*, 2016] have attempted to utilize node content information, such as textual features of each node in citation networks, for effective network representation learning. These existing works have confirmed that node content indeed provides crucial information to learn better network representations. However, as we will soon demonstrate in Section 5, these methods are mainly designed to consider consistent node content, but fail to work for user profiles in social networks. This is mainly attributed to two reasons. First, today's online social networks rely on users to manually input profile attributes, so attributes in profiles could be very

Table 1: Characteristics of information sources used for network embedding

Info. Sources	Consistency	Sparsity	Noise	Incompleteness
Structure	Medium	High	Medium	Low
Node content	High	Medium	Low	Low
User profile	Low	High	High	High

sparse, incomplete, and noisy. The values of user profile attributes are often long-tail distributed, and the values of some attributes like school or address may occur very infrequently or are simply missing. Second, different from node content such as posts and comments that are topic-centric, user profiles are depicted by user attributes on different dimensions, such as interests, or school/affiliation, and the values on these dimensions are completely distinct and inconsistent. It is thus very difficult to find useful information from user profiles that could complement network structure towards learning a joint vector representation of social networks.

In Table 1, we summarize major characteristics of information sources available for network embedding. Our analysis and empirical study confirm that user profiles are largely different from node content features, and therefore existing rich node content based network embedding methods are ineffective in handling user profiles for representation learning.

To overcome the above-mentioned difficulties, we propose a new algorithm called User Profile Preserving Social Network Embedding (UPP-SNE), which incorporates user profile information with network structure to jointly learn a vector representation of social networks. The theme of the UPP-SNE algorithm is to learn a joint embedding representation by performing a non-linear mapping on user profiles guided by network structure. In this feature reconstruction process, network structure helps filter out noisy information from user profiles and embed them into a consistent subspace, into which topology structure is seamlessly encoded to jointly learn an informative embedding representation. The interplay between user profile information and network structure enables them to complement with each other towards learning a joint network representation that preserves both network proximity and user profile affinity. The effectiveness of the UPP-SNE algorithm is validated on four real-world social networks for the tasks of node classification and clustering. Experimental results show that, UPP-SNE's representation yields superior performance to the state-of-the-art baselines.

The main contribution of this paper is threefold:

- We introduce user profile preserving social network embedding, in which user profiles that widely exist in social networks are leveraged for effective and informative social network embedding;
- We propose UPP-SNE that employs a non-linear mapping to incorporate noisy, sparse, and incomplete user profile with network structure to learn joint embeddings;
- We empirically evaluate the effectiveness of the proposed algorithm through node classification and clustering tasks on real-world social networks, showing its superior performance to the state-of-the-art baselines.

2 Related Work

In this section, we review two lines of NRL algorithms, namely network structure preserving NRL methods that are based on network structure only, and content augmented NRL methods that combine node content with network structure to enhance network representation learning.

2.1 Network Structure Preserving NRL Methods

DeepWalk [Perozzi *et al.*, 2014] is one of the pioneer works for learning node representations in networks. Following the idea of Skip-Gram [Mikolov *et al.*, 2013], DeepWalk generates node context using truncated random walks and learns node representations that allow the nodes sharing similar node context to be represented similarly. LINE [Tang *et al.*, 2015] formulates a more clear objective function to preserve the first-order proximity and the second-order proximity. GraRep [Cao *et al.*, 2015] further considers higher order proximities that describe the representation similarity between nodes sharing indirect neighbors. Very recently, SDNE [Wang *et al.*, 2016] is proposed to learn non-linear network representations by applying deep autoencoder model on node adjacent matrix and exploiting the first-order proximity as supervised information. To capture both the local and global network structure, node2vec [Grover and Leskovec, 2016] exploits biased random walks to generate context nodes, and then applies DeepWalk [Perozzi *et al.*, 2014] to learn node representations. The above NRL algorithms consider only network structure, without taking advantage of user-generated content widely available in social networks to learn more informative network representations.

2.2 Content Augmented NRL Methods

Text-associated DeepWalk (TADW) [Yang *et al.*, 2015] is the first attempt to import textual features into NRL. By proving DeepWalk is equivalent to matrix factorization, TADW incorporates textural features into network embedding through matrix factorization. TADW can be regarded as a special case of our proposed algorithm where a linear mapping is performed on node content features. TriNDR [Pan *et al.*, 2016] further exploits supervised labels to learn better node representations by modeling the inter-node relationship, node-word correlation and label-word correspondence. TADW and TriNDR are both designed for information networks with rich text content on node features (e.g., citation networks), but are ineffective for social networks with noisy user profile information. LANE [Huang *et al.*, 2017] proposes to learn three types of latent node representations via spectral techniques from the node content-level similarity matrix, network adjacent matrix and node label-level similarity matrix, and project them into a common embedding space. TriDNR and LANE primarily focus on utilizing supervised labels to enhance network representation learning. From their reported results, label information contributes more to performance gains, while the potential of node content is not sufficiently exploited.

3 Problem Definition

We consider that a social network is given as an undirected social graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$

is the set of edges. Each node $v_i \in \mathcal{V}$ is characterized by a d-dimensional feature vector $\boldsymbol{x}_i$ that describes its profile information. Social network embedding aims to map each node $v_i \in \mathcal{V}$ into a low-dimensional space, and the mapped image $\Phi(v_i)$ of node v_i is taken as the learned node representation.

The learned network representations should satisfy three properties: (1) *low-dimensional*, i.e., the dimension of the learned representations should be much smaller than the dimension of the original adjacent matrix representation, $|\mathcal{V}|$; (2) *network structure preserving*, i.e., the original network structure is fully preserved in the embedded space; and (3) *user profile preserving*, i.e., user profile information is sufficiently exploited to complement network structure for learning a joint vector representation. Note that, we focus on user profiles in this study, but our problem can be flexibly generalized to handle other modalities of node content.

4 User Profile Preserving Social Network Embedding

In this section, we first review preliminaries on DeepWalk, and then describe our proposed solution in detail.

4.1 DeepWalk

DeepWalk [Perozzi *et al.*, 2014] generalizes the Skip-Gram model from word representation learning to the learning of node representations in a network. Given a random walk node sequence $S_r = \{v_{r_1}, v_{r_2}, \ldots, v_{r_{|S_r|}}\}$ on the given network, DeepWalk learns a representation for node v_{r_i} by using it to predict its context nodes within t-window size $\{v_{r_{i-t}}, \ldots, v_{r_{i+t}}\} \backslash v_{r_i}$. This is achieved by maximizing the conditional probability of the occurrence of context nodes given the current node:

$$\max_{\Phi} \log \Pr(\{v_{r_{i-t}}, \ldots, v_{r_{i+t}}\} \backslash v_{r_i} | \Phi(v_{r_i})). \tag{1}$$

By making conditional independence assumption, the probability $\Pr(\{v_{r_{i-t}}, \ldots, v_{r_{i+t}}\} \backslash v_{r_i} | \Phi(v_{r_i}))$ is calculated as

$$\Pr(\{v_{r_{i-t}}, \ldots, v_{r_{i+t}}\} \backslash v_{r_i} | \Phi(v_{r_i})) = \prod_{j=i-t, j\neq i}^{j=i+t} \Pr(v_{r_j} | \Phi(v_{r_i})). \tag{2}$$

Above, the probability $\Pr(v_{r_j} | \Phi(v_{r_i}))$ is modeled by softmax:

$$\Pr(v_{r_j} | \Phi(v_{r_i})) = \frac{\exp(\Psi(v_{r_j}) \cdot \Phi(v_{r_i}))}{\sum_{v \in \mathcal{V}} \exp(\Psi(v) \cdot \Phi(v_{r_i}))}, \tag{3}$$

where $\Phi(v)$ and $\Psi(v)$ is the input and output representation of node v, respectively.

4.2 The UPP-SNE Algorithm

For DeepWalk, the representation of node v_i, $\Phi(v_i)$, is learned from scratch, which is independent of node content features. However, as confirmed by existing works (e.g., [Yang *et al.*, 2015; Pan *et al.*, 2016]), node content can provide crucial information to enhance network representation learning. Thus, we propose to construct node representations via a non-linear mapping from node profile features $\varphi(\boldsymbol{x_i})$:

$$\Phi(v_i) = \varphi(\boldsymbol{x}_i). \tag{4}$$

Kernel mapping is a common choice for non-linear mapping, in which $\varphi(\boldsymbol{x}_i)$ could be infinite dimensional. However, according to [Rahimi *et al.*, 2007], the infinite dimensional kernel space can be approximated by some low-dimensional feature space, which provides the potential for $\varphi(\boldsymbol{x})$ to be low-dimensional. As shown in [Rahimi *et al.*, 2007], for shift-invariant kernels, the approximated low-dimensional feature mapping $\varphi(\cdot)$ can be written as (with high probability):

$$\begin{aligned} \boldsymbol{x}_i \to \varphi(\boldsymbol{x}_i) = \frac{1}{\sqrt{m}} \Big[& \cos(\boldsymbol{\mu}_1^{\mathrm{T}} \boldsymbol{x}_i), \cdots, \cos(\boldsymbol{\mu}_m^{\mathrm{T}} \boldsymbol{x}_i), \\ & \sin(\boldsymbol{\mu}_1^{\mathrm{T}} \boldsymbol{x}_i), \cdots, \sin(\boldsymbol{\mu}_m^{\mathrm{T}} \boldsymbol{x}_i) \Big]^{\mathrm{T}}, \end{aligned} \tag{5}$$

where $\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_m\}$ are the m projection directions sampled according to the distribution from the Fourier transform of the kernel function.

In social networks, user profile attributes and network structure are highly interrelated. For example, users with similar attributes are much more likely to be friends, and groups of users with common attributes often form dense communities. Thus, in order to capture this dependency, we propose to encode network structure into the mapping $\varphi(\cdot)$ in the DeepWalk framework, such that noisy information in user profiles can be filtered out, and useful information consistent with network structure can be preserved in the embedded space.

In DeepWalk, to learn node representation $\Phi(v)$ for each node $v \in \mathcal{V}$, we need to solve the joint optimization problem:

$$\min_{\Phi} -\sum_{r} \sum_{i=1}^{|S_r|} \log \Pr(\{v_{r_{i-t}}, \ldots, v_{r_{i+t}}\} \backslash v_{r_i} | \Phi(v_{r_i})), \tag{6}$$

By making the conditional independence assumption, Eq. (6) is reformulated as

$$\min_{\Phi} -\sum_{r} \sum_{i=1}^{|S_r|} \sum_{j=i-t, j\neq i}^{j=i+t} \log \Pr(v_{r_j} | \Phi(v_{r_i})). \tag{7}$$

For convenience, Eq. (7) can be reformulated as

$$\min_{\Phi} -\sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} n(v_i, v_j) \log \Pr(v_j | \Phi(v_i)), \tag{8}$$

where $n(v_i, v_j)$ is the number of times that v_j occurs in v_i's context in a given set of random walk sequences. If v_j never occurs in v_i's context, the value of $n(v_i, v_j)$ is 0.

Following Eq. (3), the probability $\Pr(v_j | \Phi(v_i))$ is modeled by softmax. Here, we set $\Phi(v_i)$ to $\varphi(\boldsymbol{x}_i)$ and we note $\Psi(v_j)$, a $2m$-dimensional parameter vector related to v_j, as $\boldsymbol{\nu}_j$. Therefore, the probability $\Pr(v_j | \Phi(v_i))$ is formulated as

$$\Pr(v_j | \Phi(v_i)) = \frac{\exp(\boldsymbol{\nu}_j \cdot \varphi(\boldsymbol{x}_i))}{\sum_{k=1}^{|\mathcal{V}|} \exp(\boldsymbol{\nu}_k \cdot \varphi(\boldsymbol{x}_i))}. \tag{9}$$

After substituting Eq. (9) into Eq. (8), and adopting negative sampling [Gutmann and Hyvärinen, 2012], we can get the objective function of the optimization task in Eq. (8):

$$\mathcal{O} = -\sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} n(v_i, v_j) \left[\log \sigma(\boldsymbol{\nu}_j \cdot \varphi(\boldsymbol{x}_i)) + \sum_{k=1}^{K} \log \sigma(-\boldsymbol{\nu}_{N_i^k} \cdot \varphi(\boldsymbol{x}_i)) \right], \tag{10}$$

where $\sigma(\cdot)$ is the sigmoid function with $\sigma(x) = \frac{1}{1+\exp(-x)}$ and N_i^k is the index for the kth sampled negative node for node v_i. To construct node representation $\Phi(v)$ for each node $v \in \mathcal{V}$, parameters $\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_m\}$ and $\{\boldsymbol{\nu}_1, \cdots, \boldsymbol{\nu}_{|\mathcal{V}|}\}$ need to be solved by the gradient descent algorithm:

$$\begin{aligned} \boldsymbol{\mu}_s &\leftarrow \boldsymbol{\mu}_s - \eta \frac{\partial \mathcal{O}}{\partial \boldsymbol{\mu}_s}, \ \forall s = 1, \cdots, m; \\ \boldsymbol{\nu}_s &\leftarrow \boldsymbol{\nu}_s - \eta \frac{\partial \mathcal{O}}{\partial \boldsymbol{\nu}_s}, \ \forall s = 1, \cdots, |\mathcal{V}|; \end{aligned} \tag{11}$$

where the gradients are calculated as

$$\begin{aligned} \frac{\partial \mathcal{O}}{\partial \boldsymbol{\mu}_s} = &- \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} n(v_i, v_j) \left[\sigma\left(-\boldsymbol{\nu}_j \cdot \varphi(\boldsymbol{x}_i)\right) \frac{\partial \varphi(\boldsymbol{x}_i)}{\partial \boldsymbol{\mu}_s} \boldsymbol{\nu}_j \right. \\ &\left. - \sum_{k=1}^{K} \sigma\left(\boldsymbol{\nu}_{N_i^k} \cdot \varphi(\boldsymbol{x}_i)\right) \frac{\partial \varphi(\boldsymbol{x}_i)}{\partial \boldsymbol{\mu}_s} \boldsymbol{\nu}_{N_i^k} \right], \end{aligned} \tag{12}$$

$$\begin{aligned} \frac{\partial \mathcal{O}}{\partial \boldsymbol{\nu}_s} = &- \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} n(v_i, v_j) \left[\mathbf{1}_s(j) \sigma\left(-\boldsymbol{\nu}_j \cdot \varphi(\boldsymbol{x}_i)\right) \varphi(\boldsymbol{x}_i) \right. \\ &\left. - \sum_{k=1}^{K} \mathbf{1}_s(N_i^k) \sigma\left(\boldsymbol{\nu}_{N_i^k} \cdot \varphi(\boldsymbol{x}_i)\right) \varphi(\boldsymbol{x}_i) \right], \end{aligned} \tag{13}$$

where $\frac{\partial \varphi(\boldsymbol{x}_i)}{\partial \boldsymbol{\mu}_s}$ is a $d \times 2m$ Jacobian matrix. $\mathbf{1}_s(\cdot)$ is an indicator function, which is defined as

$$\mathbf{1}_s(x) = \begin{cases} 1 & \text{if } x = s, \\ 0 & \text{if } x \neq s. \end{cases} \tag{14}$$

The learning rate η is specified by line search.

Algorithm 1 The UPP-SNE Algorithm

Input: a given social network $G = (\mathcal{V}, \mathcal{E})$;
Output: node representation $\Phi(v)$ for each node $v \in \mathcal{V}$;
1: start random walk from each node by γ times;
2: count $n(u, v)$ for each node context pair $\{u, v\}$ in all random walk sequences;
3: generate K negative samples $N_i^1, \cdots, N_i^K$ for each v_i;
4: initialize $\{\boldsymbol{\mu}_s\}_{s=1}^m$ and $\{\boldsymbol{\nu}_s\}_{s=1}^{|\mathcal{V}|}$ with random numbers;
5: **repeat**
6: calculate $\frac{\partial \mathcal{O}}{\partial \boldsymbol{\mu}_s}$ for each s according to Eq. (12);
7: specify learning rate η for updating $\boldsymbol{\mu}$ by line search;
8: update $\boldsymbol{\mu}_s$ for each s according to Eq. (11);
9: calculate $\frac{\partial \mathcal{O}}{\partial \boldsymbol{\nu}_s}$ for each s according to Eq. (13);
10: specify learning rate η for updating $\boldsymbol{\nu}$ by line search;
11: update $\boldsymbol{\nu}_s$ for each s according to Eq. (11);
12: **until** convergence or a certain # of iterations
13: construct $\Phi(v)$ for each $v \in \mathcal{V}$ with $\{\boldsymbol{\mu}_s\}_{s=1}^m$;
14: **return** $\Phi(v)$ for each $v \in \mathcal{V}$.

The description of the UPP-SNE algorithm is given in **Algorithm** 1. In line 1-2, for each node acting as a starting node, γ random walk sequences are generated to calculate the statistics $n(v_i, v_j)$. In line 5-12, to solve the optimization problem of Eq. (8), gradient descent iteration is used, which contributes to the main time consumption of our algorithm. During the iteration, the most time-consuming steps are line 6 and line 9, i.e., the calculation of $\frac{\partial \mathcal{O}}{\partial \boldsymbol{\mu}_s}$ and $\frac{\partial \mathcal{O}}{\partial \boldsymbol{\nu}_s}$ for each s. The time complexity of line 6 and line 9 is $O(nnz(n) \cdot m \cdot \bar{d} \cdot K)$ and $O(nnz(n) \cdot m \cdot K)$, respectively, where $nnz(n)$ is the number of non-zero entries of $n(v_i, v_j)$ and $\bar{d}$ is the averaged number of non-zero entries of node profile feature $\boldsymbol{x}_i$. As the sparsity of matrix $n(v_i, v_j)$ and profile feature $\boldsymbol{x}_i$ is utilized, the calculation is efficient.

5 Experiments

We evaluate the validity of the proposed algorithm through node classification and clustering tasks.

5.1 Benchmark Social Networks

We perform experiments on four real-world social networks listed as follows:

Google+[1] is an ego-network of a Google+ user and nodes in this network represent the user's friends. There are 1206 nodes and 66918 links in this network. We use people's gender as class label. Each node is described by a 940-dimensional bag-of-words vector constructed from tree-structured user profiles [Leskovec and Mcauley, 2012].

Ego-Facebook[1] is an ego-network of a Facebook user. This network consists of 755 nodes and 60050 links. People's education type is used as class label. Each node is described by a 477-dimensional vector [Leskovec and Mcauley, 2012].

Hamilton and **Rochester**[2] are two of the collection of 100 US university Facebook networks [Traud *et al.*, 2012]. The two networks contain 2118 nodes, 87486 edges, and 4145 nodes, 145305 edges respectively. Each node's profile is described by 7 user attributes: student/faculty status flag, gender, major, second major/minor, dorm/house, year, and high school. We select student/faculty status flag as class label and construct a 144-dimensional, and a 235-dimensional binary feature vector for **Hamilton** and **Rochester**, respectively.

5.2 Baseline Methods

We compare our algorithm with two groups of baselines:

Network Structure Preserving NRL algorithms

- DeepWalk [Perozzi *et al.*, 2014] learns network representations with Skip-Gram model by using random walks to generate context for each node.
- LINE-1 [Tang *et al.*, 2015] is the version of LINE that models the first-order proximity.
- LINE-2 [Tang *et al.*, 2015] is the version of LINE that models the second-order proximity.
- node2vec [Grover and Leskovec, 2016] uses biased random walks to capture both local and global structure.

Content Augmented NRL algorithms

- TADW [Yang *et al.*, 2015] learns node representations by encoding node text features into the matrix factorization version of DeepWalk. It can be considered as a degraded version of UPP-SNE that uses a linear mapping.

[1] https://snap.stanford.edu/data/
[2] https://escience.rpi.edu/data/DA/fb100/

Table 2: Node classification accuracy (%) on Google+

Training Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
DeepWalk	55.24	56.51	60.85	62.17	63.41	62.30	65.86	66.08	67.06	67.39
LINE-1	52.46	54.97	58.28	59.81	60.74	61.30	62.27	63.08	63.67	63.37
LINE-2	52.14	56.24	58.78	61.18	62.49	63.33	64.83	64.83	65.81	65.50
node2vec	55.69	56.27	60.27	62.29	62.85	61.69	65.15	65.76	66.94	66.38
TADW	52.61	57.14	59.39	64.61	66.22	67.91	69.89	**70.27**	**71.89**	**71.72**
LANE	50.40	54.13	55.21	55.34	54.96	55.79	55.87	56.70	56.50	57.25
UPP-SNE	**61.86**	**69.99**	**69.20**	**69.78**	**70.33**	**70.95**	**70.37**	69.11	68.92	68.46

Table 3: Node classification accuracy (%) on Hamilton

Training Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
DeepWalk	84.64	86.76	87.63	89.40	89.67	90.11	90.37	90.81	90.79	90.51
LINE-1	80.68	83.16	83.98	86.94	87.36	88.22	89.11	89.56	89.89	89.90
LINE-2	80.23	82.41	83.03	86.78	86.79	87.87	88.36	89.05	89.15	89.55
node2vec	85.16	**87.46**	88.01	90.18	90.14	90.62	90.89	91.45	91.38	91.20
TADW	81.57	84.00	85.27	87.54	87.48	88.48	88.56	89.92	89.11	89.33
LANE	79.54	80.04	79.96	81.43	81.16	82.27	82.94	83.62	83.34	84.34
UPP-SNE	**85.35**	86.89	**89.14**	**90.68**	**90.76**	**91.71**	**91.79**	**92.07**	**92.51**	**92.41**

Table 4: Node classification accuracy (%) on Rochester

Training Ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
DeepWalk	82.49	83.58	84.14	84.40	85.08	85.43	85.65	85.71	85.66	86.13
LINE-1	81.41	81.58	82.19	82.90	83.62	84.21	84.67	84.75	84.87	85.47
LINE-2	81.33	81.29	81.80	82.43	83.09	83.71	84.16	84.29	84.43	85.15
node2vec	82.24	82.93	83.29	83.99	84.57	84.99	85.52	85.25	85.33	86.02
TADW	80.68	80.83	81.44	81.70	81.98	82.33	82.83	83.16	83.32	84.26
LANE	80.91	80.90	80.84	80.98	81.10	81.09	81.11	81.22	81.17	81.48
UPP-SNE	**83.28**	**84.88**	**85.89**	**86.14**	**87.20**	**87.83**	**87.61**	**87.82**	**88.33**	**88.20**

- LANE [Huang *et al.*, 2017] learns three kinds of node latent representations by preserving node pairwise similarity defined by node content, links, and labels, and project them into a common embedding space. Here, we only use the version that does not consider labels.

5.3 Parameter Settings

For UPP-SNE, we set the number of random walks per node γ as 40, the walk length l as 40, the window size t as 10, and the number of iterations as 40. All baselines use default parameters as reported. The dimension of learned node representations m is set as 128 for all the algorithms.

5.4 Node Classification

We first verify the effectiveness of the UPP-SNE algorithm for multi-class node classification. To make fair comparisons, we vary the training ratio from 1% to 10% by an increment of 1%. For each training ratio, we randomly split the training set and test set for 10 times and report the averaged accuracy.

Tables 2-4 report the classification accuracy of all the algorithms on Google+, Hamilton and Rochester, where the best results are **bold-faced**. We can see that, in most cases, UPP-SNE outperforms both network structure preserving baselines and content augmented baselines. On Google+, UPP-SNE's representations achieve the best accuracy on all training ratios except for 8%, 9% and 10%. On Hamilton, UPP-SNE performs best on 9 out of 10 training ratios. On Rochester, UPP-SNE beats all baselines on all training ratios.

The performance gains of the UPP-SNE algorithm over the structure preserving NRL baselines indicate that, with user profile information being properly exploited, better social network representations can be learned. This confirms the usefulness of user profiles in learning social network representations and the capacity of UPP-SNE in effectively capturing useful information from user profiles to complement network structure for learning better user representations.

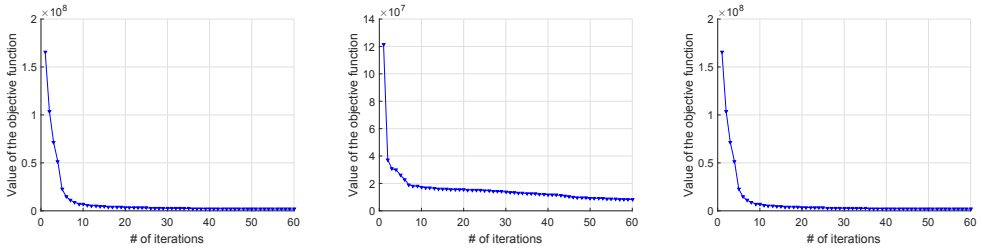
As reflected on Tables 3 and 4, on Hamilton and Rochester, TADW performs even worse than the only network structure preserving NRL algorithms. It implies that, on the two Facebook networks, structural relationships are more indicative if noisy user profile information is not properly exploited. This observation also proves that the linear mapping used by TADW cannot effectively filter noisy information from user profiles, thereby degrading its performance.

We can also see that, LANE suffers from the same problem in exploiting user profiles to learn social network representations. LANE learns latent user profile and network representations separately, and project them to get the final representations. Because the learning of latent user profile representations and latent network representations is uncorrelated, the interplay between user profile and network structure is not well captured, resulting in unsatisfactory performance.

5.5 Node Clustering

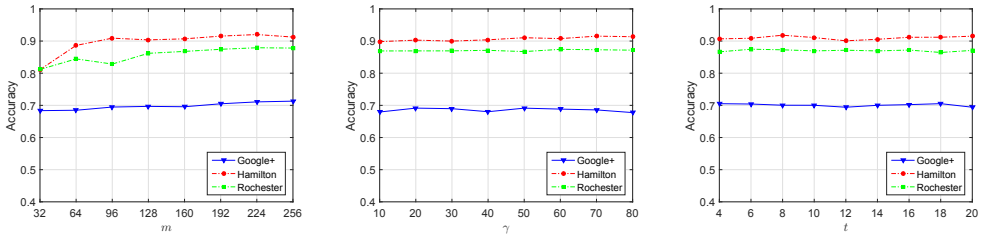
To evaluate our method on node clustering tasks, we apply k-means to the learned representations of nodes and use accuracy and NMI [Strehl and Ghosh, 2003] to assess the quality of the clustering results. We repeat the clustering process 20 times to reduce the sensitivity of k-means to the initial randomly selected centroids, and report the averaged results.

Figure 3 reports the clustering results on Google+ and Ego-Facebook. As we can see, UPP-SNE consistently yields the best clustering results. On Google+, compared with the second best results, UPP-SNE achieves 200% improvement



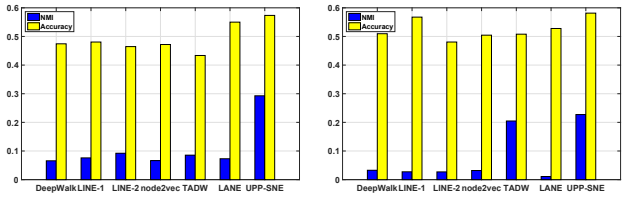
(a) Google+ (b) Hamilton (c) Rochester

Figure 1: Convergence of the objective function



(a) m (b) γ (c) t

Figure 2: The effect of parameters m, γ, and t



(a) Google+ (b) Ego-Facebook

Figure 3: The performance of node clustering

over LINE-2 on NMI, and 4% improvement over LANE on accuracy, demonstrating the superior performance of our method. On Ego-Facebook, UPP-SNE significantly outperforms TADW, yielding 14% and 11% performance gains on accuracy and NMI. This again confirms UPP-SNE's better capability in handling and incorporating noisy user profiles to learn social network representations.

5.6 Convergence Analysis

We also conduct experiments to investigate the convergence property of solving the optimization problem in Eq. (8). We vary the number of iterations from 1 to 60 and plot the corresponding value of the objective function on Google+, Hamilton, and Rochester, as shown in Figure 1. We can see that, our algorithm can achieve fast convergence within only 10 iterations on the three networks.

5.7 Parameter Sensitivity Study

Experiments are performed to analyze UPP-SNE's sensitivity to three parameters: (1) m: the dimension of learned node representations; (2) γ: the number of random walks from each node; and (3) t: the window size for collecting context nodes. In turns, we fix any two of the three parameters and study the effect of the third one on the classification accuracy. Figure 2 shows the accuracy of multi-class node classification on Google+, Hamilton, and Rochester, when the training ratio is set as 5%. We can see that the performance of UPP-SNE is stable with respect to different values of the parameters.

6 Conclusion

In this paper, we proposed a user profile preserving social network embedding method. We argued that, although a handful of methods exist to leverage node content for network representation learning, node content and user profile are largely different; while the former is topic-centric, the latter is noisy, inconsistent, highly incomplete, and uninformative. As a result, existing node content augmented network representation learning methods are ineffective to leverage node profile information. Accordingly, we proposed to use a nonlinear mapping to augment network structure and node profile information, in order to jointly learn an informative representation. Experiments on node classification and clustering tasks demonstrate the superior performance of the proposed method in learning effective social network representations.

Acknowledgements

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

[Bhuyan *et al.*, 2014] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014.

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

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.

[Gutmann and Hyvärinen, 2012] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(2):307–361, 2012.

[Huang *et al.*, 2017] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *Proceedings of 10th ACM International Conference on Web Search and Data Mining*, pages 731–739, 2017.

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

[Lü and Zhou, 2011] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[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 *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.

[Pan *et al.*, 2016] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 1895–1901, 2016.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710. ACM, 2014.

[Rahimi *et al.*, 2007] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pages 1177–1184, 2007.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, 2008.

[Strehl and Ghosh, 2003] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2003.

[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 the 24th International Conference on World Wide Web*, pages 1067–1077. ACM, 2015.

[Traud *et al.*, 2012] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

[Wang *et al.*, 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234. ACM, 2016.

[Yang *et al.*, 2013] Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *Proceedings of the 13th IEEE International Conference on Data Mining*, pages 1151–1156. IEEE, 2013.

[Yang *et al.*, 2015] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 2111–2117, 2015.