JMLR: Workshop and Conference Proceedings 63:1–16, 2016 ACML 2016

Non-Linear Smoothed Transductive Network Embedding with Text Information

Weizheng Chen[1] CWZ.PKU@GMAIL.COM
Xia Zhang[1] ZHANGXIA9403@GMAIL.COM
Jinpeng Wang[2] JINPWA@MICROSOFT.COM
Yan Zhang[1] ZHYZHY001@GMAIL.COM
Hongfei Yan[1] YHF1029@GMAIL.COM
Xiaoming Li[1] LXM.AT.PKU@GMAIL.COM
[1] *School of Electronic Engineering and Computer Science, Peking University, Beijing, China*
[2] *Microsoft Research, Beijing, China*

Editors: Robert J. Durrant and Kee-Eung Kim

Abstract

Network embedding is a classical task which aims to map the nodes of a network to low-dimensional vectors. Most of the previous network embedding methods are trained in an unsupervised scheme. Then the learned node embeddings can be used as inputs of many machine learning tasks such as node classification, attribute inference. However, the discriminant power of the node embeddings maybe improved by considering the node label information and the node attribute information.

Inspired by traditional semi-supervised learning techniques, we explore to train the node embeddings and the node classifiers simultaneously with the text attributes information in a flexible framework. We present **N**on-**L**inear **S**moothed **T**ransductive **N**etwork **E**mbedding (NLSTNE), a transductive network embedding method, whose embeddings are enhanced by modeling the non-linear pairwise similarity between the nodes and the non-linear relationships between the nodes and the text attributes. We use the node classification task to evaluate the quality of the node embeddings learned by different models on four real-world network datasets. The experimental results demonstrate that our model outperforms several state-of-the-art network embedding methods.

1. Introduction

Data that has the network architecture characteristic, such as social network and document network, has been studied by the computer science research community for a long time. Nowdays, statistical machine learning techniques are widely used in the network analysis area for various important tasks, say, node classification(Yang et al., 2016), network visualization(Tang et al., 2016), user alignment (Liu et al., 2016) and link prediction(Tang et al., 2015b). However, a real network is usually represented as a high dimensional sparse adjacent matrix which makes it difficult to apply traditional machine learning algorithms on the network data.

Network embedding, also known as network representation learning, is a fundamental problem in the network analysis area. The key idea is to project each node into a low-

dimensional vector space. Thus the node embeddings can be used as features for other data mining tasks. Many primeval network embedding methods are based on spectral factorization, such as LLE (Roweis and Saul, 2000), Laplacian Eigenmap (Belkin and Niyogi, 2001) and DGE (Chen et al., 2007). They have graceful mathematical properties but very large computational complexity which leads to inefficiency or unavailability to deal with large networks.

Recently, motivated by the success of the unsupervised distributed representation learning techniques in the natural language processing area, several novel network embedding methods have been proposed to learn distributed dense representations for networks. Perozzi et al. proposes DeepWalk (Perozzi et al., 2014), in which the node sequences are sampled from a network and fed to the Skip-Gram (Mikolov et al., 2013) model as pseudo sentences. Tang et al. later presents LINE (Tang et al., 2015a), in which one edge is sampled with several negative noisy edges in each iteration to optimize an objective function that preserves the first-order proximity or second-order proximity. Though the above two models can handle very large networks, they still have weaknesses. In the real world, the nodes in the networks are usually associated with the label or the category information, such as the conference that a paper belongs to in a paper citation network, the company a person works in the LinkedIn social network. Because the label information is not utilized in the unsupervised framework, the distinguishability of the learned embedding is limited. Furthermore, both DeepWalk and LINE ignore the rich text attributes in real networks. For instance, the users in Twitter and Facebook social network are associated with plenty user generated contents which can provide great potential to improve the expressive power of the node embeddings.

To incorporate the label information into the network embedding process, semi-supervised learning techniques, especially transductive learning methods, have been adapted to enhance the discriminative power of the learned embeddings. LSHM (Jacob et al., 2014) and MMDW (Tu et al., 2016) are two representative methods among them. They both can learn node embeddings and train a classifier on the labeled nodes simultaneously. But LSHM use a simple linear regularization smoothing term to encode the pairwise structure information of a network, which could not accurately depict the non-linear topological property (Luo et al., 2011). And MMDW is optimized in a matrix decomposition framework, which is not suitable for large networks. In addition, the performance of these two models is limited since the text information is not considered.

In this paper, we propose NLSTNE, an efficient semi-supervised network embedding method. Three components of NLSTNE, i.e., a non-linear smoothing term for the non-linear network structure information, a non-linear loss function for the text information and Linear Support Vector Machines for the label information, are naturally coupled and trained alternately. Note that even if one of the three information is not available, our model can still make use of the rest two. Therefore, NLSTNE is a general network embedding method which is not depend on the text information.

In summary, this paper has the following three major contributions:

1. We propose a novel transductive network embedding model, namely NLSTNE. NLSTNE is able to combine the label information and the non-linear structure information together to learn discriminative node features in a low-dimensional space. Signif-

icant memory saving can be acquired since NLSTNE can achieve its best performance when the length of the embedding vector is 10.

2. Our model can be further enhanced by incorporating the text attribute information into the embedding process. The semantic similarity of nodes can be preserved in the learned node embeddings by modeling the non-linear relationships between the nodes and the words.

3. To illustrate the superiority of the proposed model, the learned embeddings are evaluated within the multi-class node classification task on four real datasets with or without the text information. The results show that our model outperforms other state-of-the-art baselines significantly. Specifically, our model achieves more than 17% improvement over the most competitive baseline MMDW (Tu et al., 2016) when only 10% nodes are labeled. We also provide a parameter sensitivity study to demonstrate the robustness of our model.

The rest of this paper is organized as follows. Section 2 gives a discussion of the related work. Section 3 formally defines our research problem. Section 4 introduces our proposed model and its implementation in details. Section 5 presents the experimental results of node classification and parameter tuning. Finally we conclude in Section 6.

2. Related Work

Our work is primarily related to network embedding models. According to whether the label information is used, we break up those models into two separate categories.

The first category is unsupervised network embedding models. Usually these models learn the node embeddings by optimizing an objective function designed to preserve some certain properties of the network structure. For example, DGE (Chen et al., 2007) embeds the nodes of a directed network to a vector space by preserving the inherent pairwise node relations measured by transition probability and the stationary distribution. In recent years, DeepWalk(Perozzi et al., 2014) adopts Skip-Gram (Mikolov et al., 2013), a popular distributed word representation learning method, to the network representation learning task. It has been shown that DeepWalk is actually equivalent to factoring a matrix whose entry is the logarithm of the average probability that a node can randomly walk to another node in a fixed number of steps. Tang et al. later propose LINE(Tang et al., 2015a), which optimizes a carefully designed objective function that preserves both the first-order proximity and second-order proximity. Cao et al. present GraRep (Cao et al., 2015) , which integrates the global structural information of the network into the learning process by optimizing k-step loss functions in a matrix factorization framework. Walklets (Perozzi et al., 2016) considers the offsets between the nodes observed in a random walk to learn a series of representations. Like DeepWalk, all these models can be explained as a neural matrix factorization framework whose input is usually a function of the adjacent matrix. How to incorporate the text information of nodes into the embedding process has also been studied. The most recent work is TADW (Yang et al., 2015), in which SVD is performed on the TF-IDF matrix of all documents to get robust text features. TADW applies inductive

matrix completion (Natarajan and Dhillon, 2014) to leverage the text features into the matrix factorization style DeepWalk.

The second category is semi-supervised network embedding models. Semi-supervised network embedding models are usually tuned for the node classification task by taking the label information into consideration. Usually, a semi-supervised network embedding model is an extension of an unsupervised network embedding model (Jacob et al., 2014; Li et al., 2016; Tu et al., 2016). For instance, The objective function of DDRW (Discriminative Deep Random Walk) (Li et al., 2016) is a linear combination of two parts. The first part is the objective function of DeepWalk. And the second part is the objective function of a $l2$ regularized support vector machine trained on the labeled nodes. Furthermore, it is still not well studied how to utilized text information of nodes into a semi-supervised network embedding framework. Thus our motivation is to enhance node embeddings with the rich text attributes information and the label information simultaneously.

3. Problem Formulation

Given a partially labeled network $G = (V, E)$, V is the set of nodes and $E \subseteq (V \times V)$ is the set of edges. For an edge $e_{i,j} \in E$, w_{ij} is its weight. If there is no edge between v_i and v_j, we set $w_{ij} = 0$ .We use $L = \{v_1, ..., v_{|L|}\}$ and $U = \{v_{|L|+1}, ..., v_{|L|+|U|}\}$ to represent the labeled nodes and the unlabeled nodes respectively. Here, $V = L \cup U$ and $|V| = |L| + |U|$. Let K be the number of labels in the network, we use a vector $\boldsymbol{y_i}$ to encode the label information of node v_i. If v_i has the label $k \in \{1, ..., K\}$, $y_i^k = 1$, otherwise $y_i^k = -1$.

The target of network embedding is to learn a continuous vector representation $\boldsymbol{z_i} \in \mathbb{R}^d$ for each node v_i, where $d \ll |V|$. In most previous related works, the node embeddings are used to solve the task of predicting labels of the unlabeled nodes. If we adopt an unsupervised network embedding method, the first step is to learn embeddings of all nodes without considering the known labels of the labeled nodes. Then the embeddings of the labeled nodes are used to train a classifier and the embeddings of the unlabeled nodes are fed to the classifier to get the predicted results. However, in a transductive network embedding method, the embeddings of all nodes and a classifier trained on the labeled nodes will be optimized alternatively. Thus, the label information is directly used to enhance the distinguishability of the embeddings of the labeled nodes, which in turn improves the quality of the unlabeled nodes' embeddings. Finally, the learned classifier can be used to predict the labels of the unlabeled nodes.

4. Our Model

In this section, we first describe our non-linear smoothing regularization term. Next, we describe how to model the non-linear relationships between nodes and attributes. Then, we introduce how to utilize the label information by training a classifier. Finally, we propose a novel transductive network embedding method, NLSTNE. The model training and complexity analysis are also discussed.

4.1. Non-Linear Network Smoothing

To preserve the network structure information in the learned node embeddings, LSHM, a representative semi-supervised network embedding model, adopts the following linear smoothing regularization term:

$$O_{ls} = \sum_{(i,j)\in E} w_{ij} ||\boldsymbol{z_i} - \boldsymbol{z_j}||^2. \tag{1}$$

Here, O_{ls} is inspired by Laplacian Eigenmaps (Belkin and Niyogi, 2001), which is a classical unsupervised dimension reduction algorithm. By minimizing O_{ls}, the embeddings of nodes linked by an edge will get closer in the measure of Euclidean distance in the hidden space. To preserve the pairwise similarity in the network, the norm's square of the difference of the vectors of the linked nodes is used as the penalty term in O_{ls}. However, previous work (Luo et al., 2011) stated that the network structure may be ***highly non-linear***.

We also consider the pairwise similarity, namely the weight of the edge between two nodes. Given two nodes v_i and v_j, whose embedding vectors are $\boldsymbol{z_i}$ and $\boldsymbol{z_j}$ respectively, we adopt the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$ to model the probability that an edge between them is observed:

$$\begin{aligned} p_1(v_i, v_j) =& \sigma(\boldsymbol{z_i}^T \boldsymbol{z_j}) \\ =& \frac{1}{1+\exp(-\boldsymbol{z_i}^T \boldsymbol{z_j})}, \end{aligned} \tag{2}$$

where $p_1(\cdot,\cdot)$ is a probability distribution over all node pairs. According to the observed network structure information, the corresponding empirical probability of $p_1(\cdot,\cdot)$ is defined as:

$$\hat{p_1}(v_i, v_j) = \frac{w_{ij}}{Z}, \tag{3}$$

where $Z = \sum_{(i,j)\in E} w_{ij}$ is a normalization constant. If the learned node embeddings capture enough network structure information, the two distributions $p_1(\cdot,\cdot)$ and $\hat{p_1}(\cdot,\cdot)$ should be close to each other. By adopting Kullback-Leibler divergence (Kullback and Leibler, 1951) as the distance metric between two probability distributions, we define the following loss function to model the network structure:

$$\begin{aligned} O_{network} =& D_{KL}(\hat{p_1}||p_1) \\ =& \sum_{(i,j)\in E} \hat{p_1}(i,j) \log \frac{\hat{p_1}(i,j)}{p_1(i,j)} \\ =& \sum_{(i,j)\in E} \hat{p_1}(i,j) \log \hat{p_1}(i,j) - \sum_{(i,j)\in E} \hat{p_1}(i,j) \log p_1(i,j) \\ =& \sum_{(i,j)\in E} \hat{p_1}(i,j) \log \hat{p_1}(i,j) - \frac{1}{Z} \sum_{(i,j)\in E} w_{ij} \log p_1(i,j). \end{aligned} \tag{4}$$

After omitting those constants from the above function, we get our non-linear smoothing regularization term:

$$
\begin{aligned}
O_{nls} &= -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j) \\
&= \sum_{(i,j)\in E} w_{ij} \log \left[1 + \exp\left(-\boldsymbol{z_i}^T \boldsymbol{z_j}\right)\right], \\
&= \sum_{(i,j)\in E} w_{ij} O_{nls}^{ij}.
\end{aligned} \tag{5}
$$

where O_{nls}^{ij} is the non-linear smoothing loss for the edge e_{ij}. The properties of the learned node embeddings by minimizing the above non-linear smoothing term O_{nls} and the linear smoothing term O_{ls} are very different. For two nodes v_i and v_j, if the value of w_{ij} is large, the Euclidean distance of $\boldsymbol{z_i}$ and $\boldsymbol{z_j}$ will be small when O_{ls} is minimized. However, in the case of minimizing O_{nls}, the inner product of $\boldsymbol{z_i}$ and $\boldsymbol{z_j}$ will be large which results in a large probability defined in Eqn. (2). Thus we believe that the proposed non-linear smoothing term O_{nls} can better capture non-linear network structure information than the linear smoothing term O_{ls}.

4.2. Text Attributes Modeling

In the real-world networks, the nodes often have rich text attributes (namely words). For example, users in online social network such as Twitter and Facebook publish a large number of posts, and papers in a citation network are associated with the corresponding text content. It is necessary to uncover the potential effect of the nodes text attributes in NRL process. In this part, we will build a new a component to model the non-linear relationship between nodes and attributes.

Assuming the vocabulary of text attributes is A, by throwing out the attributes order information in each node's text content, we can transfer those text contents into a bipartite network $B = \{V \cup A, E_B\}$, in which we put the nodes on one side and the attributes on the other side. For an edge $b_{ij} \in E_B$, its weight f_{ij} is the frequency that the attribute a_j appears in the text content of the node v_i. To learn node embeddings from this bipartite network, the probability that the context attribute a_j is observed given a node v_i is defined as the following softmax function:

$$
p_2(a_j|v_i) = \frac{\exp(\boldsymbol{h_j}^T \cdot \boldsymbol{z_i})}{\sum_{k=1}^{|A|} \exp(\boldsymbol{h_k}^T \cdot \boldsymbol{z_i})}, \tag{6}
$$

where $\boldsymbol{h_j} \in \mathbb{R}^n$ is the context embedding vector of the attribute a_j. Then according to the observed text attribute information, the corresponding empirical probability distribution of p_2 can be defined as:

$$
\hat{p_2}(a_j|v_i) = \frac{f_{ij}}{\sum_{k\in N_a(i)} f_{ik}}, \tag{7}
$$

where $N_a(i)$ is the set of attributes that are connected to v_i in B. By adopting Kullback-Leibler divergence as the distance metric again, the following objective function designed for the text information will be minimized:

$$O_t = \sum_{i \in V} \lambda_i D_{KL}(\hat{p_2}(\cdot|v_i)||p_2(\cdot|v_i)), \tag{8}$$

where $\lambda_i = \sum_{k \in N_a(i)} f_{ik}$ is the importance of d_i in B. After removing some constants, the above loss function is simplified as:

$$O_t = -\sum_{(i,j) \in E_B} f_{ij} \log p_2(a_j|v_i). \tag{9}$$

It is computationally expensive to directly optimize Eqn. (9) since we need to iterate all the attributes in A. Hence, we can use the negative sampling technique (Mikolov et al., 2013) to reduce the computation complexity. Thus O_t is rewritten as:

$$\begin{aligned} O_t &= -\sum_{(i,j) \in E_B} f_{ij} \left\{ \log \sigma(\boldsymbol{h_j}^T \cdot \boldsymbol{z_i}) + \sum_{m=1}^{M} E_{a_n \sim P_n(a)} \left[\log \sigma(-\boldsymbol{h_n}^T \cdot \boldsymbol{z_i}) \right] \right\} \\ &= \sum_{(i,j) \in E_B} f_{ij} \left\{ O_r^{ij} + \sum_{m=1}^{M} E_{a_n \sim P_n(a)} O_n^{in} \right\} \end{aligned} \tag{10}$$

where M is the number of negative edges and $P_n(a) \propto (\sum_{i=1}^{|V|} f_{ia})^{0.75}$ is the noisy attribute distribution. For each real edge $b_{ij} \in E_B$, we sample M negative noisy edges b_{in} according to $P_n(a)$. We set $O_r^{ij} = -\log \sigma(\boldsymbol{h_j}^T \cdot \boldsymbol{z_i})$ and $O_n^{in} = -\log \sigma(-\boldsymbol{h_n}^T \cdot \boldsymbol{z_i})$ to represent the loss function for a real edge b_{ij} and a negative noisy edges b_{in} respectively.

4.3. Linear Support Vector Machine

In machine learning area, Linear Support Vector Machine (LSVM) is a widely used classifier which usually yields competitive and stable performance. Here, we adopt LSVM as our basic classifier. First of all, let's consider a binary classification problem for label k. By using labeled nodes as training data, the objective loss of LSVM for label k is:

$$\sum_{i=1}^{|L|} \left[max(0, 1 - y_i^k \boldsymbol{z_i}^T \boldsymbol{\theta}^k) + \lambda ||\boldsymbol{\theta}^k||^2 \right], \tag{11}$$

where $\boldsymbol{\theta^k}$ is the parameter vector of the LSVM for label k, λ is the regularization parameter. To solve the multi-class and multi-label classification problem, a typical solution is to train multiple one-vs-the-rest classifiers. Since we have K possible labels for each node, we just need to train K one-vs-the-rest LSVMs. We use the sum of the K LSVMs as the objective loss function for classification, which can be written as:

$$\begin{aligned} O_c &= \sum_{i=1}^{|L|} \sum_{k=1}^{K} \left[max(0, 1 - y_i^k \boldsymbol{z_i}^T \boldsymbol{\theta}^k) + \lambda ||\boldsymbol{\theta}^k||^2 \right], \\ &= \sum_{i=1}^{|L|} O_c^i \end{aligned} \tag{12}$$

where $O_c^i = \sum_{k=1}^{K} \left[max(0, 1 - y_i^k \boldsymbol{z_i}^T \boldsymbol{\theta}^k) + \lambda ||\boldsymbol{\theta}^k||^2 \right]$ is the classification loss of node v_i. Note that though the hinge loss function used in O_c is not differentiable, O_c still can be approximately minimized by using Stochastic Gradient Descent (SGD).

4.4. Non-Linear Smoothed Transductive Network Embedding

To encode the network structure information, the label information and the text attribute information (if available) into the unified network embeddings, we use a weighted linear combination of O_{nls}, O_c and O_t to formulate the objective loss function of the NLSTNE model:

$$
\begin{aligned}
O_{NLSTNE} =& \beta(\alpha O_c + O_{nls}) + \gamma(\alpha O_c + O_t) \\
=& \sum_{(i,j)\in E} \beta w_{ij} \Bigg\{ \alpha \mathbf{I}(i \le |L|) \sum_{k=1}^{K} \left[max(0, 1 - y_i^k \boldsymbol{z_i}^T \boldsymbol{\theta}^k) + \lambda ||\boldsymbol{\theta}^k||^2 \right] + \\
& \alpha \mathbf{I}(j \le |L|) \sum_{k=1}^{K} \left[max(0, 1 - y_j^k \boldsymbol{z_j}^T \boldsymbol{\theta}^k) + \lambda ||\boldsymbol{\theta}^k||^2 \right] + \log\left[1 + \exp\left(-\boldsymbol{z_i}^T \boldsymbol{z_j}\right)\right] \Bigg\} \\
& + \sum_{(i,j)\in E_B} \gamma f_{ij} \Bigg\{ \alpha \mathbf{I}(i \le |L|) \sum_{m=1}^{M+1} \sum_{k=1}^{K} \left[max(0, 1 - y_i^k \boldsymbol{z_i}^T \boldsymbol{\theta}^k) + \lambda ||\boldsymbol{\theta}^k||^2 \right] \\
& - \log \sigma(\boldsymbol{h_j}^T \cdot \boldsymbol{z}_i) - \sum_{m=1}^{M} E_{a_n \sim P_n(a)} \left[\log \sigma(-\boldsymbol{h_n}^T \cdot \boldsymbol{z_i}) \right] \Bigg\} \\
=& \sum_{(i,j)\in E} \beta w_{ij} \left\{ \alpha O_c^i + \alpha O_c^j + O_{nls}^{ij} \right\} + \sum_{(i,j)\in E_B} \gamma f_{ij} \left\{ \alpha \sum_{m=1}^{M+1} O_c^i + O_r^{ij} + \sum_{m=1}^{M} E_{a_n \sim P_n(a)} O_n^{in} \right\}
\end{aligned}
\tag{13}
$$

where α, β and γ are three tunable trade-off parameters, $\mathbf{I}(x)$ is an indicator function. Note that the value of γ is either 0 or 1. If the text attribute information is available, we set $\gamma = 1$. Otherwise we set $\gamma = 0$ to make NLSTNE a general transductive semi-supervised network embedding model. Furthermore, the value of β is one of $\{0.25, 0.5, 1.0, 2.0, 4.0\}$ and λ is either 0.01 or 0.001.

Once the classifier parameters and the node embeddings are learned by minimizing O_{NLSTNE}, the labels of those unlabeled nodes can be predicted by feeding their embeddings to the learned LSVMs.

4.5. Training and Complexity Analysis

As shown in Algorithm 1, we give a brief introduction to the optimization framework of NLSTNE-T. In general, SGD method has been used to update the parameters of the classifier and the embeddings alternately. $\boldsymbol{\Theta}$ is a $d \times K$ matrix whose kth column is the classifier parameter vector $\boldsymbol{\theta}^k$. η is the learning rate of SGD.

Line 3-7 shows how the label information and the non-linear network structure information are incorporated together. In each iteration, we first sample an edge $e_{ij}$from E according to the weight of the edges. For each node of the edge, if it is labeled, the classifier parameters and its node embedding will be updated by using the UpdateLSVM algorithm

defined in Algorithm 2. Then regardless of whether it is labeled or not, its embedding will be updated according to the non-linear smoothing loss O_{nls}^{ij}.

In a similar way, Line 8-17 shows how the label information and the text attribute information are incorporated together. We sample a real edge b_{ij} from E_B according to the weight of b_{ij}. Then we successively update $\boldsymbol{\Theta}$, $\boldsymbol{z_i}$ and $\boldsymbol{h_j}$. Finally we sample M negative noisy edge b_{in} and update the corresponding parameters.

Algorithm 1 Stochastic Gradient Descent for NLSTNE-T

input : a partially labeled network G and its corresponding text bipartite network B
output: node embeddings $\boldsymbol{z}$, attribute embeddings $\boldsymbol{h}$, classifier parameters $\boldsymbol{\Theta}$

1 initialize all node, attribute embeddings and classifier parameters randomly
2 **for** *A fixed number of iterations* **do**
3 sample an edge e_{ij} from E according to $w_{ij} > 0$
4 UpdateLSVM($\boldsymbol{z_i}, \boldsymbol{\Theta}, \beta\alpha\eta$)
5 UpdateLSVM($\boldsymbol{z_j}, \boldsymbol{\Theta}, \beta\alpha\eta$)
6 $\boldsymbol{z_i} \leftarrow \boldsymbol{z_i} - \beta\eta\frac{\partial O_{nls}^{ij}}{\partial \boldsymbol{z_i}} = \boldsymbol{z_i} + \beta\eta\sigma(-\boldsymbol{z_i}^T\boldsymbol{z_j})\boldsymbol{z_j}$
7 $\boldsymbol{z_j} \leftarrow \boldsymbol{z_j} - \beta\eta\frac{\partial O_{nls}^{ij}}{\partial \boldsymbol{z_i}} = \boldsymbol{z_j} + \beta\eta\sigma(-\boldsymbol{z_i}^T\boldsymbol{z_j})\boldsymbol{z_i}$
8 sample an edge b_{ij} from E_B according to $f_{ij} > 0$
9 UpdateLSVM($\boldsymbol{z_i}, \boldsymbol{\Theta}, \gamma\alpha\eta$)
10 $\boldsymbol{z_i} \leftarrow \boldsymbol{z_i} - \gamma\eta\frac{\partial O_r^{ij}}{\partial \boldsymbol{z_i}} = \boldsymbol{z_i} + \gamma\eta\sigma(-\boldsymbol{z_i}^T\boldsymbol{h_j})\boldsymbol{h_j}$
11 $\boldsymbol{h_j} \leftarrow \boldsymbol{h_j} - \gamma\eta\frac{\partial O_r^{ij}}{\partial \boldsymbol{h_j}} = \boldsymbol{h_j} + \gamma\eta\sigma(-\boldsymbol{z_i}^T\boldsymbol{h_j})\boldsymbol{z_i}$
12 **for** $m \leftarrow 1$ **to** M **do**
13 sample a negative noisy attribute a_n from $P_n(a)$
14 UpdateLSVM($\boldsymbol{z_i}, \boldsymbol{\Theta}, \gamma\alpha\eta$)
15 $\boldsymbol{z_i} \leftarrow \boldsymbol{z_i} - \gamma\eta\frac{\partial O_n^{in}}{\partial \boldsymbol{z_i}} = \boldsymbol{z_i} - \gamma\eta\sigma(\boldsymbol{z_i}^T\boldsymbol{h_n})\boldsymbol{h_n}$
16 $\boldsymbol{h_n} \leftarrow \boldsymbol{h_n} - \gamma\eta\frac{\partial O_n^{in}}{\partial \boldsymbol{h_n}} = \boldsymbol{h_n} - \gamma\eta\sigma(\boldsymbol{z_i}^T\boldsymbol{h_n})\boldsymbol{z_i}$
17 **end**
18 **end**

Algorithm 2 UpdateLSVM($\boldsymbol{z_i}, \boldsymbol{\Theta}, \eta$)

if $i \leq L$ **then**
$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \eta\frac{\partial O_c^i}{\partial \boldsymbol{\Theta}}$
$\boldsymbol{z_i} \leftarrow \boldsymbol{z_i} - \eta\frac{\partial O_c^i}{\partial \boldsymbol{z_i}}$
end

Now we analyze the time complexity of NLSTNE model. By adopting the alias method (Walker, 1974), sampling an edge from E or sampling an attribute from $P_n(a)$ takes only $O(1)$ time. In general, the probability that a random sampled node is labeled is $\frac{|L|}{|V|}$. Note that $\boldsymbol{\Theta}$ actually contains K parameter vectors, so $K+1$ vectors need to be updated if UpdateLSVM is executed. In one iteration of Algorithm 1, the average number of vectors

need to be updated is $4 + 2M + \frac{|L|}{|V|}(M + 3)(K + 1) < MK + 3M + 3K + 7$. In practice, we find that the number of iterations I should be proportional to the number of edges $|E|$, say $I = 80|E|$. Therefore, the overall time complexity of NLSTNE is $O(d|E|MK)$ when text information is available. If we have no text information, the time complexity reduces to $O(d|E|K)$. Since the time complexity of NLSTNE is not depend on $|V|$, it can scale up to very large networks.

For simplification and clarification, in the following paper, when we say NLSTNE, we refer to the reduced version of our model in which $\gamma = 0$. And when we say NLSTNE-T, we refer to the full version of our model in which the text information is used.

5. Experiments

In this section, we first conduct multi-class node classification task to evaluate different network embedding methods quantitatively on four real-world networks. Then we report the result of parameter sensitivity experiment to show the robustness of our model.

5.1. Dataset

Table 1: Statistics of the experimental datasets.

Name	Citeseer	Wiki	DBLP-3A	DBLP-4A
$\|V\|$	3,324	2,405	18,058	27,199
$\|E\|$	4,732	17,981	103,011	66,832
K	6	17	3	4
#Attributes per node	11.45	647.38	0	21.01
#Labels per node	1	1	1	1.15

We select two citation networks used in (Tu et al., 2016), Citeseer and Wiki, and two user coauthor networks, DBLP-3A (Tang et al., 2015a) and DBLP-4A (Sun et al., 2009), as our experimental datasets. Citeseer and Wiki are unweighted networks in which the directed citation relationships are transformed into the undirected edges. In Citeseer, nodes are papers and the text attributes are generated from titles and abstracts. In Wiki, nodes are Web pages and the text contents are extracted as text attributes. DBLP-3A and DBLP-4A are constructed in the same way. If two users have co-authored a paper, we add an undirected edge to link them. The weight of the edge is the number of their collaborative papers. The titles of all the paper published by one user are recognized as his or her text attributes. But text information of DBLP-3A is not available. In DBLP-3A, nodes are researchers in three different areas, namely data mining, machine learning and computer vision. In DBLP-4A, nodes are researchers in four different areas, namely data mining, machine learning, database and information retrieval.

For all these four networks, the different research areas are considered as the labels. Among them, only nodes in DBLP-4A can have multiple labels and nodes in other datasets only have one label. Some basic statistics of our datasets are given in Table 1.

5.2. Compared Algorithms

We compare the performance of the following 10 algorithms:

- DeepWalk (Perozzi et al., 2014). DeepWalk is an unsupervised network embedding method. The parameters of DeepWalk are set as follows, the sliding window size $w = 10$, the length of each node sequence t = 40, the number of node sequences for per node $\gamma = 80$.
- LINE(1st) (Tang et al., 2015a). LINE with first-order proximity, in which linked nodes will have closer representations.
- LINE(2nd) (Tang et al., 2015a). LINE with second-order proximity, in which nodes with similar neighbors will have similar representations.
- LSHM (Jacob et al., 2014). A transductive network embedding method, which uses a smoothing term to capture the linear network structure information.
- MMDW (Tu et al., 2016). A transductive network embedding method based on matrix decomposition, which also trains a Linear Support Vector Machine as its classifier.
- BOW. Each node is represented as a TF-IDF vector.
- TADW (Yang et al., 2015). A state-of-the-art unsupervised document network embedding algorithm based on inductive matrix completion.
- NLSTNE-T. Our proposed model. We set $\alpha = 1$, $\gamma = 1$, $M = 5$. For Citeseer and DBLP-4A, we set $\beta = 1$ and $\lambda = 0.01$. For Wiki, we set $\beta = 0.5$ and $\lambda = 0.001$.
- NLSTNE. A reduced version of NLSTNE-T. The parameters of NLSTNE-T are same to NLSTNE-T except that $\gamma = 0$.
- NLSUNE-T. An unsupervised reduced version of NLSTNE-T. The parameters of NLSTNE-T are same to NLSTNE-T except that $\alpha = 0$.

5.3. Node Classification

We adopt the widely used node classification task (Tang et al., 2015a; Yang et al., 2015; Tu et al., 2016) to evaluate the quality of the node embeddings learned by different models. Since NLSTNE-T, NLSTNE, LSHM and MMDW all train a max-margin LSVM as their classifiers. To make a fair comparison, the one-vs-rest LSVM is used as classifier for other models. For DBLP-4A, we adopt Micro-F1 and Macro-F1 as the evaluation metrics. For other mono-label datasets, we use accuracy as the evaluation metric. All reported results are averaged over 20 runs.

Table 2-6 show the averaged results of classification with different training ratios when the dimension d is set to 200 for all models. We mainly have two observations from those tables:

(1) When the text information is not considered, NLSTNE consistently outperforms DeepWalk, LINE, LSHM and MMDW by a noticeable margin. Compared with LSHM,

Table 2: Accuracy (%) of node classification on Citeseer.

Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	52.62	56.62	56.63	57.42	57.48	57.27	58.47	56.81	55.05
LINE(1st)	45.70	51.22	54.55	56.28	57.02	58.05	58.94	59.77	59.37
LINE(2nd)	46.68	51.23	53.36	55.41	57.55	58.14	58.37	59.00	59.04
LSHM	53.67	57.73	60.10	61.61	62.69	63.43	64.09	65.51	66.02
MMDW	54.72	59.64	62.60	64.10	65.83	68.96	69.56	69.58	69.16
NLSTNE	**55.86**	**60.82**	**64.33**	**66.88**	**68.35**	**70.58**	**73.11**	**73.33**	**74.61**
BOW	64.89	69.40	71.82	72.53	72.90	74.40	74.66	75.02	75.12
TADW	70.80	73.00	73.99	74.53	74.71	75.09	75.30	75.41	75.86
NLSUNE-T	69.53	71.04	71.98	72.51	72.65	72.52	72.75	73.16	73.49
NLSTNE-T	**71.32**	**73.62**	**74.29**	**75.33**	**76.53**	**77.44**	**77.50**	**78.97**	**79.33**

Table 3: Accuracy (%) of node classification on Wiki.

Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	55.49	58.67	60.86	63.09	64.38	65.85	66.89	67.75	67.88
LINE(1st)	54.92	61.98	64.76	66.30	67.66	68.01	68.86	67.92	69.42
LINE(2nd)	57.09	59.90	62.30	62.86	63.82	64.74	64.60	65.18	65.10
MMDW	57.25	62.01	65.04	66.67	66.89	68.23	69.22	70.18	**72.61**
LSHM	55.56	58.73	61.81	61.62	64.86	65.22	67.15	66.83	68.58
NLSTNE	**58.14**	**62.62**	**65.29**	**67.62**	**68.99**	**70.98**	**70.88**	**71.49**	71.90
BOW	72.16	76.62	77.83	79.35	80.05	80.72	80.69	81.19	81.83
TADW	72.00	75.40	77.66	78.84	79.13	80.49	80.76	79.96	80.08
NLSUNE-T	75.16	79.35	80.74	81.95	82.64	82.63	83.59	83.61	83.97
NLSTNE-T	**76.37**	**79.61**	**81.90**	**82.23**	**83.17**	**83.32**	**83.97**	**84.96**	**85.41**

Table 4: Accuracy (%) of node classification on DBLP-3A.

Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	83.18	83.64	84.06	84.10	84.46	84.16	84.51	84.28	84.94
LINE(1st)	77.93	79.77	80.16	80.46	80.61	80.75	80.74	81.08	81.35
LINE(2nd)	79.46	80.29	80.66	81.05	81.22	81.10	81.45	81.14	81.25
LSHM	82.98	85.15	85.97	87.51	88.20	89.02	89.15	89.57	90.58
MMDW	-	-	-	-	-	-	-	-	-
NLSTNE	**83.38**	**85.74**	**87.43**	**88.65**	**89.50**	**90.27**	**90.99**	**91.01**	**91.46**

NLSTNE achieves nearly 5.5%, 4.1%, 1.3%, 4.8% improvement on Citeseer, Wiki, DBLP3A and DBLP-4A in the measure of Accuracy or Micro-F1 when the training ratio is 0.5. Due to the lack of memory, the most promising baseline, MMDW, could not handle DBLP-3A and DBLP-4A on our Linux server with 64G memory. However, NLSTNE actually only need no more than 0.5G memory to process DBLP-3A and DBLP-4A on the same server.

(2) When the text information is considered, NLSTNE-T consistently outperforms all other models. NLSTNE-T achieves nearly 1.8%, 4.0%, 9.2% improvement over the most

Table 5: Micro-F1 (%) of node classification on DBLP-4A.

Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	66.85	68.03	68.57	68.79	68.85	68.95	69.10	69.31	69.50
LINE(1st)	64.06	65.73	66.39	66.77	66.67	66.89	66.81	66.96	67.20
LINE(2nd)	65.87	66.83	67.24	67.37	67.68	67.59	67.63	67.61	67.49
MMDW	-	-	-	-	-	-	-	-	-
LSHM	70.09	73.36	76.53	78.84	81.13	84.87	85.40	87.58	87.73
NLSTNE	**72.53**	**78.62**	**82.59**	**85.51**	**87.91**	**89.65**	**90.99**	**92.37**	**93.75**
BOW	78.25	81.65	83.51	85.02	86.53	87.32	88.24	89.14	89.88
TADW	80.30	82.05	82.23	82.40	82.87	83.47	83.32	82.26	82.94
NLSUNE-T	81.28	81.79	82.09	82.11	82.23	82.23	83.37	82.39	82.57
NLSTNE-T	**82.56**	**86.40**	**88.79**	**90.43**	**91.93**	**93.13**	**93.86**	**94.66**	**95.00**

Table 6: Macro-F1 (%) of node classification on DBLP-4A.

Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	52.03	54.62	59.80	60.29	61.26	65.41	65.84	66.53	68.16
LINE(1st)	60.88	62.54	63.14	63.53	63.45	63.64	63.55	63.76	64.16
LINE(2nd)	62.41	63.54	64.00	64.05	64.45	64.38	64.41	64.37	64.11
MMDW	-	-	-	-	-	-	-	-	-
LSHM	68.76	72.47	75.87	78.37	80.72	84.17	84.91	87.13	87.29
NLSTNE	**71.58**	**77.59**	**81.56**	**84.63**	**87.05**	**88.92**	**90.33**	**91.77**	**93.29**
BOW	76.89	80.41	82.47	84.06	85.62	86.49	87.53	88.42	89.02
TADW	78.91	80.64	80.89	80.95	81.43	82.08	81.99	81.13	81.83
NLSUNE-T	79.87	80.37	80.72	80.75	80.86	80.86	80.89	80.99	81.30
NLSTNE-T	**81.90**	**85.56**	**87.98**	**89.69**	**91.29**	**92.55**	**93.34**	**94.20**	**94.62**

competitive baseline TADW on Citeseer, Wiki and DBLP-4A in the measure of Accuracy or Micro-F1 when the training ratio is 0.5. Compared with TADW, NLSTNE-T has two advantages. First, the network structure information and the text information can be better balanced by choosing an appropriate β. For example, even NLSUNE-T performs better than TADW on Wiki. Second, the node embeddings learned by NLSTNE-T are more discriminative by incorporating the label information.

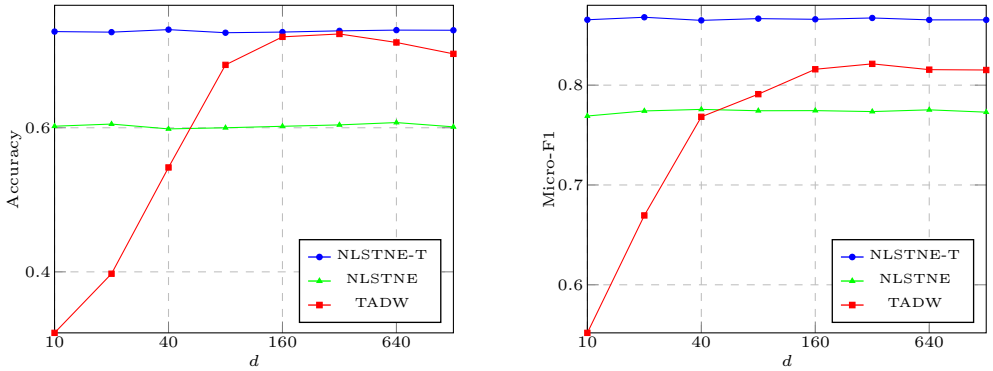
5.4. Parameter Sensitivity

We investigate the parameter sensitivity in this part to test the stability of our model. Specially, we evaluate the effect of different values of the embedding dimensions d, the trade-off parameters α and β on the classification results. when one parameter is tested, all other parameters are fixed to their default values. When the training ratio is 0.2, we show the results of parameter sensitivity on Citeseer and DBLP-4A in Figure 1 and 2.

First, we can see that NLSTNE-T and NLSTNE are not sensitive to d at all when $d \in [10, 1280]$. Actually, even $d = 10$ is enough to make our models achieve their approximate optimal performance. Meanwhile, it is not easy to determine the value of d for TADW.

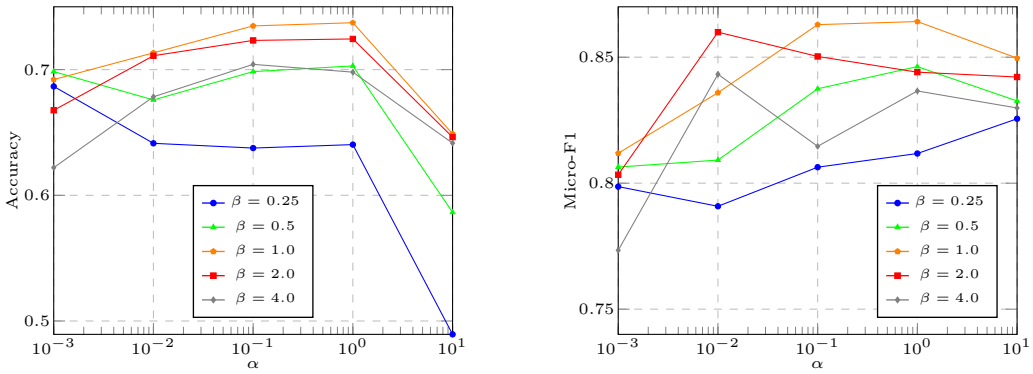
Particularly, TADW is worse than NLSTNE when $d < 40$. When d is too small or too large, the performance of TADW will tend to decrease. Overall, TADW gets its best performance when d is around 320 on both Citeseer and DBLP-4A. So the memory requirements to store the node embeddings of our models are only 3.1% of TADW. This demonstrates that significant memory savings can be acquired by incorporating the label information.

Second, we show that $\alpha = 1$ and $\beta = 1$ are appropriate initial values for NLSTNE-T. When $\alpha \in [0.001, 0.1]$, the predictive power of the node embeddings can not be guaranteed since the weight of the classification loss is too small to train the LSVMs. When $\alpha = 10$, the network structure information and the text attribute information can not be well captured since the weight of the classification loss is too large. We should give more attention to the parameter β. If the quality of the network structure information and the text attribute information are roughly comparable, $\beta = 1$ can well balance them. For example, we set $\beta = 1$ for Citeseer and DBLP-4A since the documents of the nodes in them are short texts. But we set $\beta = 0.5$ for Wiki since the documents of the nodes in it are long texts.



(a) Accuracy on Citeseer (b) Micro-F1 on DBLP-4A

Figure 1: Parameter sensitivity w.r.t. d.



(a) Accuracy on Citeseer (b) Micro-F1 on DBLP-4A

Figure 2: Parameter sensitivity w.r.t. α and β.

6. Conclusion

In this paper, we explore the problem of learning node embeddings in a transductive framework. We present NLSTNE, an efficient and effective model to learn more discriminative node embedding by utilizing the label information and the non-linear structure information. Furthermore, the text attribute information can be incorporated into NLSTNE in a flexible way. The experimental results show that our models outperform several state-of-the-art methods markedly on four benchmark datasets.

An important direction of our future work is to enhance our model by adopting deep learning techniques. Note that the embeddings updated by the SVM based classifier is shallow. A deep neural network may help us learn deep and more informative node embeddings.

7. Acknowledgments

This work is supported by 973 Program with Grant No.2014CB340400, NSFC with Grant No.U1536201 and No.61272340. Yan Zhang is supported by NSFC with Grant No. 61532001 and No. 61370054, and MOE-RCOE with Grant No. 2016ZD201. We thank the three anonymous reviewers for their insightful comments.

References

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

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.

Mo Chen, Qiong Yang, and Xiaoou Tang. Directed graph embedding. pages 2707–2712, 2007.

Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. Learning latent representations of nodes for classifying in heterogeneous social networks. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 373–382. ACM, 2014.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

Juzheng Li, Jun Zhu, and Bo Zhang. Discriminative deep random walk for network classification. 2016.

Li Liu, William K Cheung, Xin Li, and Lejian Liao. Aligning users across social networks using network embedding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 1774–1780, 2016.

Dijun Luo, Feiping Nie, Heng Huang, and Chris H Ding. Cauchy graph embedding. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 553–560, 2011.

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*, pages 3111–3119, 2013.

Nagarajan Natarajan and Inderjit S Dhillon. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics*, 30(12):i60–i68, 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.

Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Walklets: Multiscale graph embeddings for interpretable network classification. *arXiv preprint arXiv:1605.02115*, 2016.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

Yizhou Sun, Jiawei Han, Jing Gao, and Yintao Yu. itopicmodel: Information network-integrated topic modeling. In *2009 Ninth IEEE International Conference on Data Mining*, pages 493–502. IEEE, 2009.

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. International World Wide Web Conferences Steering Committee, 2015a.

Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

Jie Tang, Tiancheng Lou, Jon Kleinberg, and S Wu. Transfer link prediction across heterogeneous social networks. *ACM TOIS*, 2015b.

CunChao Tu, Weicheng Zhang, zhiyuan Liu, and Maosong Sun. max-margin deepwalk: discriminative learning of network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 3889–3895, 2016.

Alastair J Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 8(10):127–128, 1974.

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 Conference on Artificial Intelligence*, IJCAI'15, pages 2111–2117. AAAI Press, 2015. ISBN 978-1-57735-738-4.

Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.