PPNE: Property Preserving Network Embedding

Chaozhuo Li[1](✉), Senzhang Wang[2,3], Dejian Yang[1], Zhoujun Li[1], Yang Yang[1], Xiaoming Zhang[1], and Jianshe Zhou[4]

[1] State Key Lab of Software Development Environment, Beihang University, Beijing, China
{lichaozhuo,ydj1994,lizj,yangyang,yolixs}@buaa.edu.cn

[2] Nanjing University of Aeronautics and Astronautics, Nanjing, China
szwang@nuaa.edu.cn

[3] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China

[4] Capital Normal University, Beijing 100048, People's Republic of China

Abstract. Network embedding aims at learning a distributed representation vector for each node in a network, which has been increasingly recognized as an important task in the network analysis area. Most existing embedding methods focus on encoding the topology information into the representation vectors. In reality, nodes in the network may contain rich properties, which could potentially contribute to learn better representations. In this paper, we study the novel problem of property preserving network embedding and propose a general model PPNE to effectively incorporate the rich types of node properties. We formulate the learning process of representation vectors as a joint optimization problem, where the topology-derived and property-derived objective functions are optimized jointly with shared parameters. By solving this joint optimization problem with an efficient stochastic gradient descent algorithm, we can obtain representation vectors incorporating both network topology and node property information. We extensively evaluate our framework through two data mining tasks on five datasets. Experimental results show the superior performance of PPNE.

1 Introduction

Networks are ubiquitous in our daily lives and many real-life applications focus on mining information from the networks. A fundamental problem in network mining is how to learn the desirable network representations [4,22]. To address this problem, *network embedding* is presented to learn the distributed representations of nodes in the network. The main idea of network embedding is to find a dense, continuous, and low-dimensional vector for each node as its distributed representation. Representing nodes into the distributed vectors can form up a potentially powerful basis to generate high-quality node features for many data mining and machine learning tasks, such as node classification [6], link prediction [11] and recommendation [24,27].

S. Candan et al. (Eds.): DASFAA 2017, Part I, LNCS 10177, pp. 163–179, 2017.
DOI: 10.1007/978-3-319-55753-3_11

Most related works investigate the topology information for network embedding, such as DeepWalk [16], LINE [19], Node2Vec [6] and SDNE [22]. The basic assumption of these topology-driven embedding methods is that nodes with similar topology context should be distributed closely in the learned low dimensional representation space. However, in many real scenarios, it is insufficient to learn desirable node representations by purely relying on the network topology structure. For example in the social networks, it is possible that two users share very similar interests but they are not connected and share no common friends, thus their similarity on interests cannot be effectively captured by the topology based network embedding methods. In such a case, other types of information should be incorporated as the complementary content to help us learn better representations.

Usually, nodes in the network may be associated with a set of properties, such as the profiles of each user in a social network and the metadata of each paper in a citation network. Node property information is also important to measure the similarity between nodes, but are largely ignored by previous network embedding methods. For example in a social network, if two users share some common tags or tweet topics, they are very likely to be similar even if they are topologically far away from each other. Since the node properties potentially encode different types of information from the network topology, integrating them into the embedding process is expected to achieve a better performance. As the first attempt, TADW [25] incorporates the text features of nodes into network embedding process under a framework of matrix factorization. However, there are two limitations of TADW. Firstly, the very time and memory consuming matrix factorization process of TADW makes it infeasible to scale up to large networks. Secondly, TADW only considers the texts associated to each node, and it is difficult to apply TADW to handle the node properties of rich types in general.

In this paper, we propose a general network embedding framework which can effectively encode both the topology information of the network and the rich properties of the nodes. This task is difficult to address due to the following challenges. Firstly, nodes in a network may contain several types of properties, and in different networks, the types of node properties are different. It is non-trivial to model various types of properties into an unified format and utilize these property information. Secondly, although combining network topology and node properties into the embedding process is expected to achieve better performance, it is not obvious how best to do this under a general framework. There are sophisticated interactions between network topology and node properties, and it is difficult to integrate node properties into the existing topology-derived models. For example, DeepWalk and Node2Vec cannot easily handle additional information during the random walk process in a network.

To address the above challenges, we propose a general and flexible property preserving network embedding model PPNE. We formulate the learning process of property preserving network embedding as a joint optimization problem, where the topology-derived and property-derived objective functions are optimized jointly. Specifically, we propose a negative sampling based objective function to capture the topology information, which aims to maximize the likelihood of the prediction of the center node given a specific contextual node.

Besides, we extract a set of constraints according to the property similarity between each pair of nodes, and a property-derived objective function is proposed to restrict the learned representation vectors to satisfy the extracted constraints. Finally, we utilize the stochastic gradient descent (SGD) algorithm to solve this joint optimization problem.
To summarize, we make the following contributions:

– In this paper we propose and study the novel problem of property preserving network embedding, and propose a general embedding framework to effectively incorporate both network topological information and node property information into the network embedding process.
– To utilize the property similarity information, we propose two ways of extracting constraints from the property similarity matrix. A carefully designed objective function with such constraints is also proposed.
– We extensively evaluate our approach through multi-class classification and link prediction tasks on five datasets. Experimental results show the superior performance of PPNE over state-of-the-art embedding methods.

The rest of this paper is organized as follows. Section 2 summarizes the related works. Section 3 formally defines the problem of property preserving network embedding. Section 4 introduces the proposed model PPNE in details. Section 5 presents the experimental results. Finally we conclude this work in Sect. 6.

2 Related Work

Network embedding aims to learn a distributed representation vector for each node in a network, which essentially is an unsupervised feature learning process. In general, network embedding is related to the problem of *graph embedding* or *dimensionality reduction*. Most existing network embedding methods can be categorized into two broad categories: matrix factorization based and neural network based methods.

Matrix factorization based methods first express the input network with a affinity matrix in which the entries represent the relationships between nodes, and then embed the affinity matrix into a low dimensional space using matrix factorization techniques. Locally linear embedding [17] seeks a lower-dimensional projection of the input affinity matrix which preserves distances within local neighborhoods. Spectral Embedding [2] is one method to calculate the non-linear embeddings. It finds a low dimensional representation of the input data using a spectral decomposition of the graph Laplacian. Sparse random projection [10] reduces the dimensionality of data by projecting the original input space using a sparse random matrix. However, matrix factorization based methods rely on the decomposition of the affinity matrix, which is too expensive to scale efficiently to large real-world networks. Besides, the manually predefined node similarity measurements are needed to construct the affinity matrix, which can significantly affect the quality of learned representation vectors.

Recently neural network based models are introduced to solve the network embedding problem. As the first attempt, DeepWalk [16] introduces an word embedding algorithm (Skip-Gram) [14] to learn the representation vectors of nodes. Tang et al. propose LINE [19], which optimizes a carefully designed objective function that preserves both the local and global structure. Wang et al. propose SDNE [22], a deep embedding model to capture the highly non-linear network structure and preserve the global and local structures. SDNE exploits the first-order and second-order proximities to preserve the network structure. Node2Vec [6] learns a mapping of nodes to a low-dimensional space of features that maximizes the likelihood of preserving distances between network neighborhoods of nodes. TADW [25] incorporates the text features of nodes into network embedding process under a framework of matrix factorization. Compared to the matrix factorization based methods, neural network based methods are easier to generalize and own strong representation ability. However, most previous works only take the network topology information into consideration. Different from the previous typology-only works, our work aims to propose a general property preserving network embedding model which integrate the rich types of node properties in the network into the embedding process.

Finally, there is a body of works focus on the problem of node classification [8,20,21,26] or link prediction [11,13]. However, the objective of our work is totally different from these works. We aim to learn better representation vectors for nodes, while the node classification or link prediction tasks are only utilized to evaluate the quality of the embedding results.

3 Problem Definition

In this section, we formally define the studied problem. The input network G is defined as $G = (V, T, P)$, where V represents nodes in the network. The topology matrix $T \in \mathbb{R}^{|V| \times |V|}$ is the adjacency matrix of the network. P is the property similarity matrix with each entry $P_{i,j} \in [0, 1]$ denoting the property similarity score between node i and j. Here we formally define the problem of property preserving network embedding:

Definition 1. *(Property Preserving Network Embedding): Given a network G = (V, T, P), the problem of property preserving network embedding aims to learn a matrix $X \in \mathbb{R}^{|V| \times d}$, where d is the number of latent dimensions with $d \ll |V|$. Each row vector X_i in X is the embedding vector of node i. The objective of property preserving network embedding is to make the learned representation vectors explicitly preserve both the network topology and node property information.*

4 Property Preserving Network Embedding

In this section, we present the details of the proposed property preserving network embedding model PPNE. Firstly we briefly introduce the framework of PPNE. Then the topology-derived objective function and property-derived objective function are introduced separately. After that we present the joint optimization process of the above two objective functions. Finally we discuss several practical issues of the proposed model.

4.1 Framework

Figure 1 shows the framework of PPNE. One can see that each node in the network is associated with a set of properties. The first step of PPNE is to construct two matrices from the input network: the topology matrix and the property similarity matrix. Topology matrix is a 0–1 adjacency matrix which represents the connections among nodes. The property similarity matrix contains the property similarities between each pair of nodes, which is calculated by a predefined similarity measurement. Given a particular machine learning or data mining task, users can flexibly choose or design the property similarity measurement. For example, in order to serve a geographic mining task, the geography related properties (address, geographic tag) should be more important than other properties. How to calculate the node property similarities is not the focus of this paper, as it varies to different networks and applications. We assume the property similarity matrix has been given by domain experts. For the topology matrix, we utilize the random walk algorithm to generate a set of node sequences. A topology-derived objective function is proposed to capture topology information preserved in the node sequences. For the property similarity matrix, we extract a set of constraints, which ensures the embedding vectors of nodes with similar properties should be distributed closely in the learned representation space. Here we define two kinds of constraints: inequality constraints and numeric constraints. We also propose a property-derived objective function for each type of the constraints. Finally, the topology-derived and property-derived objective functions are jointly optimized sharing same parameters.

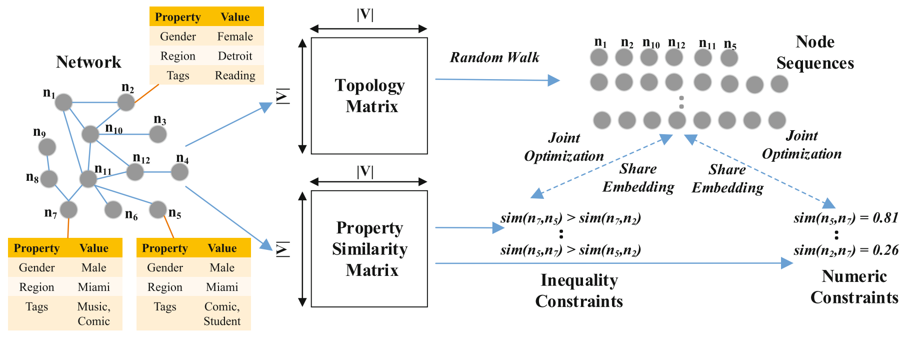


Fig. 1. Framework of PPNE.

4.2 Topology-Derived Objective Function

Following the idea of DeepWalk [16], we assume that nodes with similar topology context tend to be similar. With such an assumption, we aim to maximize the likelihood of the prediction of the center node given a specific contextual node. The contextual nodes of a center node are defined as a fixed-size window w of its previous nodes and after nodes in the node sequences generated by random walk. We propose an novel negative sampling based optimization objective function in which the node representation vectors are considered as the parameters. In the optimization procedure, the representation vectors are updated and the finally learned vectors preserve the topology information.

Firstly, the network topology matrix T is converted into a set of node sequences $\mathcal{C}$ by random walk. In a random walk iteration, for each node n in V, we generate a node sequence of length t which starts from node n. This iteration will repeat r times to generate enough sequences.

Based on the node sequences $\mathcal{C}$, we try to solve the following objective function:

$$maximize \quad D_T = \prod_{n \in \mathcal{C}} \prod_{z \in context(n)} \prod_{u \in \{\{n\} \cup NEG(n)\}} p(u|z) \tag{1}$$

Given a sampled center node n and its contextual nodes $context(n)$, $NEG(n)$ is the set of negative samples of the center node n with a predefined size ns. Nodes far away from the center node have a larger chance to be picked as the negative samples. $p(u|z)$ defines the probability of the center node u given the contextual node z. Given the contextual node z, we aim to maximize the probability of the positive samples $u \in \{n\}$, while minimize the probability of the negative samples $u \in NEG(n)$.

For each node n in V, we design two corresponding vectors: the embedding vector and the parameter vector. The embedding vector v_n is the representation of node n when it is treated as the contextual node, while the parameter vector θ_n is the representation of n when it is treated as the center node. In our model, $p(u|z)$ is defined as

$$p(u|z) = \begin{cases} \sigma(v_z^{\mathrm{T}} \theta_u), & L^n(u) = 1 \\ 1 - \sigma(v_z^{\mathrm{T}} \theta_u), & L^n(u) = 0 \end{cases}$$

in which σ is the sigmoid function:

$$\sigma(v_z^{\mathrm{T}} \theta_u) = \frac{1}{1 + e^{-(v_z^{\mathrm{T}} \theta_u)}}$$

$L^n(u)$ is an indicator function:

$$L^n(u) = \begin{cases} 1, & u \in \{n\} \\ 0, & u \in NEG(n) \end{cases}$$

Moreover, $p(u|z)$ can be represented as

$$p(u|z) = [\sigma(v_z^{\mathrm{T}} \theta_u)]^{L^n(u)} \cdot [1 - \sigma(v_z^{\mathrm{T}} \theta_u)]^{1 - L^n(u)}$$

Hence, the objective function can be rewritten as follows:

$$maximize \quad D_T = \prod_{n \in \mathcal{C}} \prod_{z \in context(n)} \prod_{u \in \{\{n\} \cup NEG(n)\}} [\sigma(v_z^{\mathrm{T}} \theta_u)]^{L^n(u)} \cdot [1 - \sigma(v_z^{\mathrm{T}} \theta_u)]^{1 - L^n(u)} \tag{2}$$

By maximizing the likelihood of prediction of positive samples and minimizing the likelihood of negative samples, the proposed objective function encodes the topology information into the representation vectors of nodes.

Compare to DeepWalk, a popular topology-derived method, the proposed model is faster and more effective. Firstly, the objective function of DeepWalk is defined as

$$maximize \quad D = \prod_{n \in \mathcal{C}} \prod_{z \in context(n)} p(z|n)$$

The hierarchical softmax method is introduced to design the probability $p(z|n)$, which reduces the computational complexity of calculating $p(z|n)$ from $O(|V|)$ to $O(\log_2 |V|)$. Hence the computational complexity of DeepWalk is $O(|\mathcal{C}| \cdot 2w \cdot \log_2 |V|)$, in which w is the window size of the contextual nodes. In the objective function (2), the computational complexity is further reduced to $O(|\mathcal{C}| \cdot 2w \cdot (ns + 1))$, in which ns is a constant number irrelevant to the size of network. Thus the model training time is significantly reduced. Secondly, by choosing the negative samples according to their distances from the center node, the rich global topology information is integrated into our model. This strategy ensures our model not only considers the local information of the contextual nodes, but also can encode the information of the nodes which have farther topological distance from the center node.

4.3 Property-Derived Objective Function

In this subsection we present the details of the property-derived objective function. In natural language processing area, SWE [12] and RC-NET [23] incorporate the semantic knowledges into the word embedding process. Inspired by the above works, we propose two ways to extract constraints from the property similarity matrix P. Based on these constraints, we introduce the property-derived objective functions.

Related works incorporate the original property features of single type into the embedding process [25]. Different from such works, in our approach the property matrix $P \in \mathbb{R}^{|V| \times |V|}$ contains the property similarity scores between each pair of nodes, which is calculated by a predefined similarity measurement. According to the requirements of targeting data mining tasks and the types of networks, users can flexibly design an appropriate property similarity measurement to cast the input network into the property similarity matrix. This strategy guarantees the proposed embedding framework can be applied on various kinds of networks and serves different types of data mining tasks.

Inequality Constraints. The first way we proposed to utilize the property similarity matrix P is to extract a set of inequalities from matrix P as the constraints. For each node n in V, according to the matrix P, we construct its most similar node set pos_n and most dissimilar node set neg_n. pos_n contains top k similar nodes of n and neg_n contains top k dissimilar nodes, where $k \ll |V|/2$. With pos_n and neg_n, we can obtain the following inequalities for node n:

$$P_{np} > P_{nq} \quad p \in pos_n, q \in neg_n$$

in which P_{np} is the property similarity score between node n and p. The final representation vectors of the corresponding nodes should satisfy the following constraint:

$$sim(v_n, v_p) > sim(v_n, v_q) \quad p \in pos_n \; q \in neg_n$$

in which v_n is the embedding vector of node n. $sim(v_n, v_p)$ is the cosine similarity between the embedding vectors of node n and p. After constructing such constraints for all the nodes, we can obtain a set of constraints $\mathcal{S}$, which contains triples in the form of $\{(i, j, k),\ sim(v_i, v_j) > sim(v_i, v_k)\}$.

Based on the constraint set $\mathcal{S}$ and the node sequences $\mathcal{C}$, we propose the following objective function to force the embedding vectors to satisfy the extracted constraints:

$$minimize \;\; D_I = \sum_{n \in \mathcal{C}} \sum_{(i,j,k) \in \mathcal{S}} I_{i,j,k}(n) \cdot f(i, j, k) \tag{3}$$

where $I_{i,j,k}(n)$ is an indicator function:

$$I_{i,j,k}(n) = \begin{cases} 1, & i = n \; or \; j = n \; or \; k = n \\ 0, & else \end{cases}$$

The function $f(.)$ is a normalization hinge loss function:

$$f(i, j, k) = max(0, sim(v_i, v_k) - sim(v_i, v_j))$$

The objective function D_I ensures that the similarity score between embedding vectors of two nodes with similar properties should be no less than the nodes with dissimilar properties. For a sampled node in the sequences, we select the inequality constraints associated with this node and judge whether the current embedding vectors of the corresponding nodes satisfy these constraints. If the constraints are satisfied, these embedding vectors will remain unchanged, otherwise they will be updated towards the direction of satisfying these constraints.

Numeric Constraints. The second way of utilizing the matrix P is to consider the property similarity scores as the numeric constraints for refining the network embedding process. The motivation is that the learned representation vectors of two nodes should be distributed closer to each other if they have similar properties, namely their property similarity score in P is high.

Similar to the generation process of the inequality constraints, according to the matrix P, for each node n in $|V|$ we select its top k similar and dissimilar node sets as the pos_n and neg_n. Nodes in the two sets own strong discrimination ability over node n. Based on the property similarity matrix P and node sequences $\mathcal{C}$, we propose the following objective function:

$$minimize \;\; D_N = \sum_{n \in \mathcal{C}} \sum_{i \in \{pos_n \cup neg_n\}} P_{ni} d(v_n, v_i) \tag{4}$$

in which $d(v_i, v_j)$ is the Euler distance function to measure the distance between the embedding vectors of node i and j: $d(v_i, v_j) = \sqrt{(v_i - v_j)^T(v_i - v_j)}$.

One can see that the optimization process of the objective function D_N is affected by the property similarity score P_{ni}, and the distances between nodes with similar properties decrease faster than that between dissimilar ones. Hence, D_N can ensure that nodes with similar properties are distributed closer in the learned embedding space.

4.4 Joint Optimization

In this subsection, we show the joint optimization process of the topology-derived and property-derived objective functions. Here we propose two types of PPNE model: $\text{PPNE}_{\text{ineq}}$ and PPNE_{num}. $\text{PPNE}_{\text{ineq}}$ aims to jointly optimize the topology-derived objective function D_T and the inequality constraint based objective function D_I. PPNE_{num} aims to jointly optimize D_T and the property-derived function D_N with numeric constraints. We utilize the SGD algorithm to solve the optimization problems. Firstly we introduce the optimization process of D_T, D_I and D_N separately, and the derivative results are utilized to jointly update the embedding vectors.

To maximize the objective function D_T in the Formula (2), we try to maximize the following log-likelihood function:

$$\log D_T = \sum_{n \in \mathcal{C}} \sum_{z \in context(n)} \sum_{u \in \{\{n\} \cup NEG(n)\}} \{L^n(u) \cdot \log[\sigma(v_z^{\mathrm{T}}\theta_u)] + [1 - L^n(u)] \cdot \log[1 - \sigma(v_z^{\mathrm{T}}\theta_u)]\}$$

Given a sample of $(n, context(n))$ in $\mathcal{C}$, with sampled z and u, we set

$$\mathcal{L} = L^n(u) \cdot \log[\sigma(v_z^{\mathrm{T}}\theta_u)] + [1 - L^n(u)] \cdot \log[1 - \sigma(v_z^{\mathrm{T}}\theta_u)] \tag{5}$$

Firstly we calculate the following partial derivative:

$$\frac{\partial \mathcal{L}}{\partial \theta_u} = [L^n(u) - \sigma(v_z^{\mathrm{T}}\theta_u)] \cdot v_z$$

Thus θ_u can be updated by

$$\theta_u = \theta_u + \eta[L^n(u) - \sigma(v_z^{\mathrm{T}}\theta_u)] \cdot v_z \tag{6}$$

where parameter η is the learning rate. In Formula (5), θ_u and v_z are symmetric, so v_z can be updated as

$$v_z = v_z + \eta \sum_{u \in \{\{n\} \cup NEG(n)\}} [L^n(u) - \sigma(v_z^{\mathrm{T}}\theta_u)] \cdot \theta_u \tag{7}$$

Then we show the optimization process of D_I in Formula (3). For convenience, we use s_{np} to represent $sim(v_n, v_p)$. Given a sampled $(n, context(n))$ in $\mathcal{C}$, firstly we calculate the partial derivative of D_I:

$$\frac{\partial D_I}{\partial v_n} = \sum_{(i,j,k)\in S} f' \cdot (\delta_{ik}(n)\frac{\partial s_{ik}}{\partial v_n} - \delta_{ij}(n)\frac{\partial s_{ij}}{\partial v_n}) \tag{8}$$

in which $\delta_{ij}(n)$ are defined as

$$\delta_{ij}(n) = \begin{cases} 1, & i = n \ or \ j = n \\ 0, & others \end{cases}$$

and

$$f' = \begin{cases} 1, & s_{ij} < s_{ik} \\ 0, & s_{ij} \geq s_{ik} \end{cases}$$

The partial derivatives in Formula (8) can be easily calculated. For example, given a constraint contains n and assume $i = n$, we can get

$$\frac{\partial s_{ij}}{\partial v_n} = \frac{\partial s_{nj}}{\partial v_n} = -\frac{s_{nj} v_n}{|v_n|^2} + \frac{v_j}{|v_n||v_j|}$$

For the sample $(n, context(n))$, the embedding vectors of node n will be updated as follows:

$$v_n = v_n - \beta \cdot \eta \sum_{(i,j,k)\in S} \frac{\partial D_I}{\partial v_n} \tag{9}$$

in which β is the balance parameter to control the weight of the node properties in the embedding process.

Here we present the optimization process of D_N in Formula (4). Given a sampled $(n, context(n))$, we calculate the partial derivative of D_N:

$$\frac{\partial D_N}{\partial v_n} = \sum_{i\in\{pos_n \cup neg_n\}} P_{ni} \cdot (v_n - v_i)[(v_n - v_i)^T (v_n - v_i)]^{-\frac{1}{2}}$$

The embedding vector of node n will be updated as follows:

$$v_n = v_n - \beta \cdot \eta \sum_{i\in\{pos_n \cup neg_n\}} \frac{\partial D_N}{\partial v_n} \tag{10}$$

Finally, Algorithm 1 shows the joint optimization process of $\text{PPNE}_{\text{ineq}}$ model. For PPNE_{num} model, we only need to modify line 14 to update embedding vector following Formula (10).

Algorithm 1. PPNE$_{\text{ineq}}$

```
   Input:
1      Network $G(V, T, P)$ ;
2      Embedding size $d$;
   Output:
3      Embedding matrix $X \in \mathbb{R}^{|V| \times d}$ ;
4  for node $n$ in $V$ do
5      initialize embedding vector $v_n \in \mathbb{R}^{1 \times d}$;
6      initialize parameter vector $\theta_n \in \mathbb{R}^{1 \times d}$;
7  end
8  node sequences $\mathcal{C}$ = RandomWalk() ;
9  for $(n, context(n))$ in $\mathcal{C}$ do
10     $\#Topology - derived\ Objective\ Function$ ;
11     update embedding vectors following Formula (7) ;
12     update parameter vectors following Formula (6) ;
13     $\#Property - derived\ Objective\ Function$ ;
14     update embedding vector following Formula (9) ;
15 end
16 for $i = 0; i < |V|; i++$ do
17     $X_i = v_i$
18 end
19 return $X$
```

4.5 Discussion

We discuss several issues of the PPNE model in detail.

Choice of the Measurements. We choose the cosine similarity measurement to measure the similarity between embedding vectors, and the Euler distance to measure the distance between embedding vectors. The proposed model is still effective with these simple and popular measurements, which can better show the generality of our model.

Property Similarity Matrix. The property similarity matrix of the input network is the basis of the proposed model. For a large network, the calculation process of the pairwise similarities between nodes seems to be time consuming. However there are several effective strategies which can significantly reduce the time cost. Firstly, we can precompute the norm of each vector and store it using a lookup table. Secondly, this process can be implemented easily in parallel. These improvements has been implemented in a popular machine learning toolkit: scikit-learn[1]. With scikit-learn, it takes only several hours to construct the similarity matrix for the largest network in our experiments.

[1] http://scikit-learn.org/stable/.

5 Experiments

In this section, firstly we introduce the datasets and baseline methods used in this work. Then we thoroughly evaluate our proposed methods through two classic data mining tasks on four paper citation networks and one social network. Finally we analyze the quantitative experimental results and investigate the sensitivity across parameters.

5.1 Experiment Setup

Datasets. In order to thoroughly evaluate the proposed methods, we conduct experiments on four paper citation networks and one social network with different scale of nodes. Table 1 shows the detailed information of the five datasets. The four paper citation networks are Citeseer[2], Cora (see Footnote 2), PubMed (see Footnote 2) [18] and DBLP[3] [9]. In the paper citation networks, nodes refer to papers and links refer to the citation relationships among papers. Papers are classified into several categories according to the belonged domains. In Citeseer, Wiki and PubMed networks, each paper has abstract as its property, and in DBLP dataset, each paper has properties like title, authors, publication venue and abstract. Google+ (see Footnote 3) is a social network in which nodes refer to users and links represent friend relationships among users. Each user has gender, job title, university and workplace as his properties. The institution of each user is considered as his category. We select top 6 popular institutions as the final categories. For the networks with single type of node property (Citeseer, Cora and PubMed), the property similarity matrix contains the pairwise cosine similarity scores between nodes. For the networks with richer and more complex node properties (DBLP and Google+), we calculate the cosine similarity between nodes over each type of properties separately, and then weighted linearly combine them. The weights are tuned in a few random sampled instances.

Embedding Methods. We compare PPNE with the following baseline methods:

- **DeepWalk:** DeepWalk [16] is a topology-only network embedding method, which introduces the Skip-Gram algorithm to learn the node representation vectors.
- **LINE:** LINE [19] is a popular topology-only network embedding method, which considers the first-order and second-order proximities information.
- **Property Features:** In this method nodes are represented by the property features.
- **Naive Combination:** We simply concatenate the vectors from both Property Features and DeepWalk as the final representation vectors.
- **TADW:** TADW [25] incorporates the text features of each node into the embedding process under a framework of matrix factorization.

2 http://linqs.cs.umd.edu/projects/projects/lbc/index.html.
3 https://snap.stanford.edu/data/index.html.

– **PPNE$_{\mathbf{ineq}}$** : PPNE$_{\text{ineq}}$ is the PPNE model with the inequality constrains.
– **PPNE$_{\mathbf{num}}$** : PPNE$_{\text{num}}$ is the proposed PPNE model with the numeric constrains.

Table 1. Statistics of the datasets

Data	Nodes	Links	Categories
Citeseer	3,312	4,732	6
Wiki	2,405	17,981	11
PubMed	19,717	44,338	3
DBLP	244,021	4,354,534	9
Google+	107,614	13,673,453	6

Parameter Setup. For all datasets, the dimension of the learned representation vector is set to $d = 160$. In DeepWalk method, parameters are set as window size $w = 10$, walks per node $r = 80$ and walk length $t = 40$. In LINE method, the parameters are set as follows: $negative = 5$ and $samples = 10$ million. In Property Features method, we reduce the dimension of node property features to 160 via SVD [7] algorithm. In TADW method the parameters are set to the same as given in the original paper. In PPNE method, the number of negative samplings $ns = 5$, the balance parameter $\beta = 0.3$, learning rate $\eta = 0.1$, walks per node $r = 80$, walk length $t = 40$.

5.2 Multi-class Classification

We utilize the representation vectors generated by various network embedding methods to perform multi-class node classification task. The representation vector of each node is treated as its feature vector, and then we use a linear support vector machine model [3] to return the most likely category. The classification model is implemented using scikit-learn. For each dataset, a portion (T_r) of the labeled nodes are randomly picked as the training data, and the rest of nodes are the test data. We repeat this process 10 times, and report the average performance in terms of classification accuracy.

Table 2 shows the classification performance on four datasets. Here "-" means TADW can not handle large networks due to its very time and memory consuming process of matrix factorization. From Table 2, one can see PPNE consistently outperforms other baseline methods. For Citeseer dataset, PPNE$_{\text{ineq}}$ achieves the best performance and beat the best baseline TADW by 5%. In Wiki dataset, PPNE$_{\text{ineq}}$ beat baselines by 3%. PPNE$_{\text{ineq}}$ improves the classification performance by 5% on DBLP dataset and PPNE$_{\text{num}}$ beat the best baseline by nearly 6% on Google+ dataset. Besides, the improvement over TADW is statistically significant (sign test, p-value < 0.05) on Citeseer and Wiki datasets. PPNE$_{\text{ineq}}$ extracts the inequalities between nodes from the property similarity

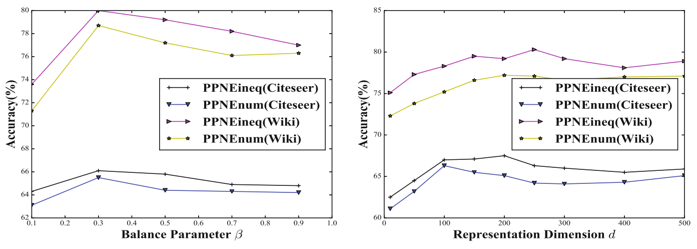
Table 2. Classification performance (Accuracy) on four datasets.

Methods	Citeseer			Wiki			DBLP			Google+		
T_r	10%	20%	30%	10%	20%	30%	1%	2%	3%	1%	2%	3%
DeepWalk	47.8	53.5	56.5	56.9	61.8	64.0	63.0	64.8	65.7	55.9	56.9	57.4
LINE	41.2	45.8	49.5	57.6	59.4	63.2	61.2	62.8	63.8	53.7	54.6	55.1
Property features	53.4	55.8	58.4	58.1	63.3	65.4	68.6	69.8	71.2	59.8	60.9	61.2
Naive combination	54.1	56.5	60.5	64.4	69.3	72.3	69.4	71.2	72.9	61.8	62.7	64.2
TADW	55.9	58.5	61.8	71.0	74.9	77.3	-	-	-	-	-	-
PPNE$_{\text{ineq}}$	**60.4**	**63.2**	**66.1**	**74.5**	**77.7**	**80.0**	**76.2**	**77.8**	**79.2**	67.2	69.1	70.3
PPNE$_{\text{num}}$	58.5	62.7	65.5	71.4	75.0	76.7	75.5	76.9	78.7	**68.7**	**70.9**	**71.8**

matrix, which is a robust method to represent the similarity information. As a more delicate method, the optimization process of PPNE$_{\text{num}}$ is affected by the numeric similarity scores, which essentially introduces the degree of inequalities between nodes. If these degrees match the label information, such as in Google+, PPNE$_{\text{num}}$ performs better than PPNE$_{\text{ineq}}$, otherwise it may introduce noise into PPNE$_{\text{num}}$ as shown on Citeseer, Wiki and DBLP datasets.

The experimental results demonstrate the effectiveness of the proposed embedding methods. By incorporating the network topology and node property information into a unified embedding framework, the quality of the learned representation vectors are improved. Meanwhile, PPNE performs consistently better when training data is small.

Parameter Sensitivity Analysis. PPNE has two major parameters: dimension d and the balance parameter β. We fix the training proportion to 30% and test the classification accuracies with different d and β. We let β varies from 0.1 to 0.9 and d varies from 10 to 500. Figure 2 shows the classification performances with different β and d on Citeseer and Wiki datasets. With the increase of β, the classification accuracy first increases and then decreases. When we increase the dimension d, the classification accuracy first increases and then keeps stable. It shows that, PPNE achieves the best performance when β varies within a reasonable range and d is larger than a specific threshold.



(a) Balance Parameter β. (b) Representation Dimension d.

Fig. 2. The classification performance w.r.t the balance parameter β and dimension d.

5.3 Link Prediction

Given a snapshot of the current network, the link prediction task refers to predicting the edges that will be added in the future time [1,13]. Link prediction can show the predictability of different network embedding methods. To process the link prediction task, a portion of existing links (50%) are removed from the input network. Based on the residual network, node representation vectors are learned by different embedding methods. Node pairs in the removed edges are considered as the positive samples. We also randomly sample the same number of node pairs that are not connected as the negative samples. Positive and negative samples form a balanced data set. Given a node pair in the samples, the cosine similarity score is calculated according to their representation vectors. Area Under Curve (AUC) [5] is used to evaluate the consistency between the labels and the similarity scores of the samples. We also choose *Common Neighbors* as a baseline method because it has been proved as an effective method [11,15].

Table 3. Link prediction performance (AUC score)

Method	Citeseer	PubMed
LINE	0.725	0.751
DeepWalk	0.743	0.78
Common neighbors	0.691	0.714
TADW	0.757	0.792
$\text{PPNE}_{\text{ineq}}$	**0.791**	**0.846**
PPNE_{num}	0.783	0.812

Table 3 shows the experimental results. One can see that PPNE outperforms other embedding methods. Compare to TADW, $\text{PPNE}_{\text{ineq}}$ improves the AUC score by 4% in Citeseer and 5% in PubMed, which demonstrates the effectiveness of PPNE in learning good node representation vectors for the task of link prediction.

6 Conclusion

This paper proposes a general network embedding model PPNE to incorporate both network topology information and node property information. We formulate the learning of property preserving network embedding as a joint optimization problem. Firstly we propose the topology-derived objective function and property-derived objective function, and then the above objective functions are optimized jointly sharing the same parameters. Experimental results on the multi-class classification and link prediction tasks over five datasets demonstrate the effectiveness of PPNE.

Acknowledgement. This work was supported by Beijing Advanced Innovation Center for Imaging Technology (No. BAICIT-2016001), the National Natural Science Foundation of China (Grand Nos. 61370126, 61672081, 61602237, U1636211, U1636210), National High Technology Research and Development Program of China (No. 2015AA016004), the Fund of the State Key Laboratory of Software Development Environment (No. SKLSDE-2015ZX-16).

References

1. Al Hasan, M., Zaki, M.J.: A survey of link prediction in social networks. In: Aggarwal, C.C. (ed.) Social Network Data Analytics, pp. 243–275. Springer, New York (2011)
2. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)
3. Chang, C.-C., Lin, C.-J.: Libsvm: a library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) **2**(3), 27 (2011)
4. Chang, S., Han, W., Tang, J., Qi, G.J., Aggarwal, C.C., Huang, T.S.: Heterogeneous network embedding via deep architectures. In: The ACM SIGKDD International Conference, pp. 119–128 (2015)
5. Fawcett, T.: An introduction to roc analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)
6. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: The ACM SIGKDD International Conference (2016)
7. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev. **53**(2), 217–288 (2010)
8. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016)
9. Leskovec, J., Krevl, A.: SNAP datasets: Stanford large network dataset collection, June 2014. http://snap.stanford.edu/data
10. Li, P., Hastie, T.J., Church, K.W.: Very sparse random projections. In: The ACM SIGKDD International Conference, pp. 287–296 (2006)
11. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Assoc. Inf. Sci. Technol. **54**(7), 1345–1347 (2007)
12. Liu, Q., Jiang, H., Wei, S., Ling, Z.-H., Hu, Y.: Learning semantic word embeddings based on ordinal knowledge constraints. In: ACL, pp. 1501–1511 (2015)
13. Menon, A.K., Elkan, C.: Link prediction via matrix factorization. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011. LNCS, vol. 6912, pp. 437–452. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23783-6_28
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
15. Newman, M.E.: Clustering and preferential attachment in growing networks. Phys. Rev. E **64**(2), 025102 (2001)
16. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: The ACM SIGKDD International Conference, pp. 701–710. ACM (2014)
17. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)

18. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93 (2008)
19. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: International Conference on World Wide Web, pp. 1067–1077 (2015)
20. Tang, L., Liu, H.: Leveraging social media networks for classification. In: CIKM, pp. 1107–1116 (2009)
21. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Min. Knowl. Discov. **23**(3), 447–478 (2011)
22. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: The ACM SIGKDD International Conference (2016)
23. Xu, C., Bai, Y., Bian, J., Gao, B., Wang, G., Liu, X., Liu, T.-Y.: Rc-net: a general framework for incorporating knowledge into word representations. In: CIKM, pp. 1219–1228 (2014)
24. Yan, S., Xu, D., Zhang, B., Zhang, H.-J., Yang, Q., Lin, S.: Graph embedding and extensions: a general framework for dimensionality reduction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 40–51 (2007)
25. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: International Conference on Artificial Intelligence, pp. 2111–2117 (2015)
26. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: ICML (2016)
27. Zhang, H., Li, Z., Chen, Y., Zhang, X., Wang, S.: Exploit latent dirichlet allocation for one-class collaborative filtering. In: CIKM, pp. 1991–1994 (2014)