BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 03/08 pp248–260
Volume 2, Number 4, December 2019
DOI: 10.26599/BDMA.2019.9020009

Network Representation Based on the Joint Learning of Three Feature Views

Zhonglin Ye, Haixing Zhao*, Ke Zhang, Zhaoyang Wang, and Yu Zhu

Abstract: Network representation learning plays an important role in the field of network data mining. By embedding network structures and other features into the representation vector space of low dimensions, network representation learning algorithms can provide high-quality feature input for subsequent tasks, such as network link prediction, network vertex classification, and network visualization. The existing network representation learning algorithms can be trained based on the structural features, vertex texts, vertex tags, community information, etc. However, there exists a lack of algorithm of using the future evolution results of the networks to guide the network representation learning. Therefore, this paper aims at modeling the future network evolution results of the networks based on the link prediction algorithm, introducing the future link probabilities between vertices without edges into the network representation learning tasks. In order to make the network representation vectors contain more feature factors, the text features of the vertices are also embedded into the network representation vectors. Based on the above two optimization approaches, we propose a novel network representation learning algorithm, Network Representation learning algorithm based on the joint optimization of Three Features (TFNR). Based on Inductive Matrix Completion (IMC), TFNR algorithm introduces the future probabilities between vertices without edges and text features into the procedure of modeling network structures, which can avoid the problem of the network structure sparse. Experimental results show that the proposed TFNR algorithm performs well in network vertex classification and visualization tasks on three real citation network datasets.

Key words: network representation learning; network feature mining; embedding learning; link prediction; matrix factorization

1 Introduction

Network Representation Learning (NRL) can be visually interpreted as the procedure that gives each vertex in the networks a low-dimensional representation vector containing local or global features of the network vertices. The representation vectors obtained by NRL algorithms have this kind of property that adjacent vertices in the networks have a relatively closer space distance in the network representation vector space, and conversely, it has a relatively farther space distance. Because the network representation vectors obtained by NRL algorithms can reflect the network structures and other information, so the representation vectors can be used to do some machine learning tasks[1–4], such as link prediction, network vertex classification, recommendation system, and visualization.

In the beginning of NRL, it uses spectral information

• Zhonglin Ye, Haixing Zhao, Ke Zhang, and Yu Zhu are with Key Laboratory of Tibetan Information Processing, the College of Computer, Qinghai Normal University, Xining 810008, China. E-mail: zhonglin_ye@foxmail.com; h.x.zhao@163.com; hasanli@163.com; zhuyu@qhu.edu.cn.
• Zhaoyang Wang is with the College of Mathematics and Statistics, Qinghai Normal University, Xining 810008, China. E-mail: wzy02130822@163.com.
* To whom correspondence should be addressed.
Manuscript received: 2018-11-13; revised: 2019-04-15; accepted: 2019-04-16

of the networks to represent the low-dimensional representation vectors. Subsequently, network representation learning based on neural network attracts more and more attention because it can be used for large-scale network feature coding tasks. NRL based on neural network originates from the word representation learning based on neural network, which is shortly called as network representation learning or distributed network representation learning.

DeepWalk[5] is the most classical algorithm in network representation learning. The most classical algorithm of word representation learning is Word2Vec[6]. There is an inheritance relationship between these two kinds of algorithms. We have mentioned that the network representation learning algorithms based on neural network originate from word representation learning algorithms based on neural network. Specifically, DeepWalk is improved based on Word2Vec algorithm. There exists a great similarity between these two kinds of algorithms, namely, the underlying models and optimization processes of these two algorithms are the same, the difference is that DeepWalk obtains the random walk sequences of network vertices as "sentences" through the random walk strategy. In addition, the input of DeepWalk and Word2Vec algorithms is "sentence", the output is a representation vector of lower dimension. DeepWalk algorithm has shown its excellent performance in large-scale network vertex classification, visualization, link prediction, and other tasks. Therefore, the improvement algorithms of network representation learning based on DeepWalk are subsequently proposed. This kind of improvement is generally based on two ways, one is based on network structures[7–10], and the other is based on joint representation learning[11–19]. Of course, there are also some improved algorithms for specific network types[20, 21].

Word2Vec adopts Continuous Bag-Of-Words model (CBOW) or Skip-Gram to model the relationships between words, as well as it adopts Negative Sampling or Hierarchical Softmax to accelerate the model training speed. Skip-Gram model using Negative Sampling is also called as SGNS model for short. Similarly, DeepWalk uses the same underlying models and optimization approaches as Word2Vec. We can find in Ref. [22] that SGNS in the language model is equivalent to implicitly factorizing the Shifted Positive Point Mutual Information matrix (SPPMI) between words. Subsequently, we can find in Refs. [23, 24] that SGNS in the network model is equivalent to implicitly factorizing the transition probability matrix M between network vertices, and $M = (P + P^2)/2$. Consequently, TADW[25] algorithm and MMDW[26] algorithm optimize the network representation learning procedure based on the theory of matrix factorization. TADW algorithm first introduces Inductive Matrix Completion (IMC)[27] to jointly learn the network representations using network structure features and vertex text features. Both TADW and MMDW adopt the idea of matrix factorization to optimize the network representation learning tasks. The difference is that MMDW adopts Singular Value Decomposition (SVD)[28] to factorize the transition probability matrix M between network vertices, while TADW adopts IMC algorithm to factorize the transition probability matrix M between network vertices. In addition, TADW algorithm uses text features to compensate for the sparse network structures, while MMDW uses network vertex tags to compensate for the sparse network structures based on max-margin theory[29].

Although many existing network representation learning algorithms optimize the network representation learning tasks based on the network structure features, vertex texts, tags, communities, and other features. However, some important indexes and conclusions of link prediction have not yet been introduced into the network representation learning algorithms so as to optimize the network representation learning tasks. As we all know, link prediction algorithm can predict the future link probabilities between vertices without edges in the networks, and it can also evaluate the link certainty degrees of existing edges, which is also known as the link weights of edges. In addition, vertices in the network also contain a large amount of text contents. In social networks, the texts of vertices are the personal information, comments, published contents, and so forth. In citation networks, the texts of vertices are usually the titles and abstracts of the papers. In order to intuitively show the principle of the algorithm proposed in this paper, the following explanatory diagram is given. The specific results are shown in Fig. 1.

In Fig. 1, we show a simple network structure consisting of multiple vertices and edges. We enlarge the link relations of some areas and set the vertex number of the local network as 1, 2, 3, and 4. These four vertices have three edges, and then we calculate

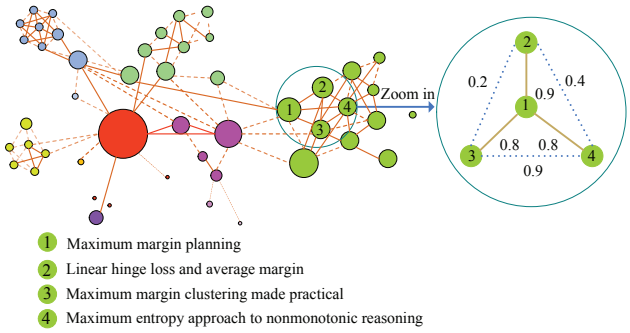


Fig. 1 Diagram of TFNR algorithm.

the corresponding weights of these three edges. The weights of existing edges in the network can be considered as the similarity or correlation between vertices. In addition, there are three dotted lines in the local network, which do not exist in the original network in Fig. 1. We define the dotted line as the future links of vertices, and give the weights of three dotted lines, the weight is the future link probabilities between vertices without edges. For example, the weight between vertex 1 and vertex 3 is 0.8, and the weight between vertex 1 and vertex 4 is also 0.8. Therefore, the future link probability between vertex 3 and vertex 4 is 0.9. Similarly, the future link probability between vertex 2 and vertex 3 is 0.2, and the future link probability between vertex 2 and vertex 4 is 0.4. Finally, we give the text contents (paper titles) of these four vertices, and we find that vertex 1 and vertex 3 are both papers related to "max-margin".

In order to introduce future link probabilities between vertices without edges and text feature into the network representation learning tasks, we put forward a novel Network Representation learning algorithm based on the joint optimization of Three Features, the algorithm is named as TFNR for short. As can be seen from Fig. 1, there exists the possible future links and existent edges between vertices in the network. The TFNR algorithm predicts the future evolution of the networks through the link prediction algorithm, and calculates the future link probabilities between vertices without edges. Although this kind of probability is obtained based on the existing network structures, it can implicitly guide the network representation learning model to carry out the training in the direction of future evolution results, so that the learnt network representation vectors obtained by this method contain the future influence factors. In addition, TFNR algorithm integrates the text features of network vertices into the network representation vectors, so that vertices with more common words will have a closer space distance in the network representation vector space. In order to embed the above two feature optimizations into the network representation learning tasks at the same time, TFNR algorithm introduces the IMC algorithm, where the essence of IMC algorithm is one kind of matrix factorization algorithm, namely, IMC algorithm learn constraint features from other two auxiliary feature matrices while factorizing the target feature matrix. Consequently, the network representation vectors obtained by TFNR algorithm contain the network structure feature factors, the future link probability factors, text feature factors, etc.

2 Our Method

2.1 Formalization

In this paper, we define the network as $G = (V, E)$, where V is a set composed of vertex v, and E is also a set composed of edge e. The input of NRL is the network G, and the input of some NRL algorithms based on the deep neural network is the spectrum or adjacency matrix of the network. The output of NRL algorithms is usually the low-dimensional network representation vector $r_v \in \mathbb{R}^k$, where k represents the size of column dimension of the network representation vectors. In this paper, we use the network representation vectors $r_v \in \mathbb{R}^k$ obtained by NRL algorithms to conduct network vertex classification, visualization, and case analysis tasks, which can verify the network representation learning performance of the proposed TFNR algorithm.

2.2 Future link probabilities between vertices without edges

Link prediction algorithm is mainly used to predict the future link probabilities between vertices without edges[30, 31]. By sorting all the future link probabilities, we can get the vertex pairs that are most likely to build edges in the next moment. Link prediction algorithm is mainly used in social networks to predict future interactions between friends, and it can also be used in recommendation systems for commodity recommendation[32, 33]. The most popular method of link prediction algorithm is based on matrix factorization.

The TFNR algorithm proposed in this paper introduces the future link probabilities between vertices without edges into the network representation learning tasks. Therefore, we first need to consider how to

measure the future link probabilities between vertices without edges and which algorithm do we need to consider to measure the future link probabilities between vertex pairs without edges.

For the above two problems, we use link prediction algorithm to calculate the future link probabilities between vertices without edges. In the processes of computing the future link probabilities, we only consider the existing structures of the networks to calculate the future link probabilities between vertex pairs, and we do not use the method of combining structures and text features to calculate the future link probabilities between vertex pairs. The main reason is that TFNR algorithm has incorporated the text features of the network vertices into the network representations. Therefore, we can only measure the performance improvement and influence of the future link probability without text features on network representation learning tasks. In order to find a desirable link prediction algorithm, we implement the existing 21 types of link prediction algorithms in the experimental sections, and evaluate the prediction performance of each link prediction algorithm on three real citation network datasets. Finally, we adopt the Matrix-Forest Index (MFI) algorithm to measure the future link probabilities between vertex pairs without edges. Because the MFI algorithm has shown its excellent prediction performance on three real citation network datasets. MFI algorithm can obtain the future link probabilities and link weights between vertices by the following matrix operation:

$$M^{\mathrm{MFI}} = (I+L)^{-1} \tag{1}$$

where I denotes the identity matrix with the size of $|V| \times |V|$, L is the laplacian matrix of the network G.

Note that Eq. (1) can simultaneously calculate the weights of existing links and the future link probabilities between vertex pairs without edges, thus, matrix M^{MFI} is composed of two different kinds of property values. So, we need to divide the weights of existing edges and the future link probabilities between vertices without edges from matrix M^{MFI}.

Suppose that the adjacency matrix of network G is A, C is defined as the adjacency matrix of complementary graph of network G. Consequently, the weights of existing edges can be calculated by

$$M^{\mathrm{weight}} = M^{\mathrm{MFI}} . \cdot A \tag{2}$$

The future link probabilities between vertices without edges can be calculated by

$$M^{\mathrm{probability}} = M^{\mathrm{MFI}} . \cdot C \tag{3}$$

The symbol " .·" denotes the product form of matlab programm grammar between two matrices, where the values of the same positions are multiplied.

2.3 Structure feature matrix construction

DeepWalk can use CBOW or Skip-Gram to model the relationships between vertex pairs, and use Negative Sampling or Hierarchical Softmax to accelerate the model training speed. Therefore, DeepWalk algorithm can be realized by two models and two optimization approaches. In addition, Skip-Gram model based on Negative Sampling optimization is called as SGNS for short, and its objective function is

$$L(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{-t \leqslant j \leqslant t, j \neq 0} \log \Pr(v_{j+i} | v_i) \tag{4}$$

where

$$\Pr(v_j | v_i) = \frac{\exp(v'_j \cdot v_i)}{\sum_{v \in V} \exp(v' \cdot v_i)} \tag{5}$$

In Eqs. (4) and (5), t denotes the number of context vertices before and after the current central vertex v_i. v_i denotes the network representation vector of current vertex v_i, v_j denotes the network representation vector of context vertex v_j. The symbol "·" denotes a dot product between two network representation vectors.

In Ref. [24], Yang and Liu found that the essence of DeepWalk based on SGNS model is to implicitly factorize the structural feature matrix of the networks. In the structural feature matrix, the value of each element is

$$M_{ij} = \log \frac{[e_i(P + P^2 + \cdots + P^t)]_j}{t} \tag{6}$$

where P is the transition matrix of network G, $P_{ij} = 1/d_i$ if $(i, j) \in E$, and $P_{ij} = 0$ otherwise. d_i is the vertex degree of vertex i . In vector e_i, the value of i-th term is 1, and the remaining terms are set to 0.

The structural feature matrix M constructed by Eq. (6) has higher computation complexity. Moreover, the matrix M calculated by Eq. (6) contains a large number of non-zero elements after logarithmic operation. Therefore, Yang and Liu[24] suggested that Eq. (7) can replace Eq. (6).

$$M = \frac{P + P^2}{2} \tag{7}$$

We even can define $M = P$ in some dense networks. In the TFNR algorithm, Eq. (7) is used to construct the structural feature matrix M of the network G. Because

P can be defined as the first order feature matrix of the network G, P^2 can be defined as the second order feature matrix of the network G. Therefore, the structural feature matrix of network G constructed by Eq. (7) contains both first-order features and second-order features between network vertices.

2.4 Text feature matrix construction

Various data can be transformed into network structure form to display and mine important data. The network reflects the relationships between different objects by edges, which is also the most important features of the networks. However, vertices in the networks also contain rich text features except the edge relationships. For example, the text contents of a vertex are the comments made by other users and comments made by users in social networks, while the followee and follower relationships among users are the edge relationships. In this paper, we mainly use citation networks to verify the network representation learning performance of TFNR algorithm. Therefore, the texts of the vertex are mainly the titles and abstracts of the papers in citation networks. As we all know that the title of the paper is a high summary of the whole paper, and the abstract contains the technology and algorithm information used in the paper. Therefore, if the relationships between vertices are analyzed only based on the vertex contents in the citation networks, the important structural features of the citation networks can also be mined.

In the TFNR algorithm, we first delete all stopwords in the texts of the citation network vertices, and then we delete all these words that the word frequency is less than 10. We then put the rest of the words into an array as a text feature dictionary. The words in this dictionary are the column header of text feature matrix T. The row header of the text feature matrix is the vertices in the network G. The rules for constructing the text feature matrix are as follows: if the column header of the matrix appears in the texts of network vertex, we set the value of this position in the text feature matrix to 1, otherwise, we set the value of this position to 0. The element values of the first row of the text feature matrix are the text feature transformation of the first vertex.

The column dimension of the text feature matrix constructed here is equivalent to the size of the text feature dictionary T. Therefore, the text feature matrix is a matrix with higher dimension, and the matrix contains a large number of zero elements, which results in a larger computation cost in the factorization procedure of the matrix. It is well known that the dimensionality reduction algorithm based on matrix factorization can remove the redundant features between different objects, and retain the features of optimal discrimination in lower dimensional space at the same time. Therefore, the text feature matrix can only be used after dimensionality reduction in the TFNR algorithm.

2.5 TFNR algorithm

We have known that the essence of DeepWalk based on SGNS model is to factorize the network structure feature matrix M. In order to explain the factorization process in detail, we give the following diagram in Fig. 2.

According to Eq. (7), DeepWalk implicitly factorizes network structure feature matrix M, where M is a transition probability matrix, and the elements in matrix M are composed of the reciprocals of the degree values of network vertices. As shown in Fig. 2, DeepWalk aims to factorize matrix $M \in \mathbb{R}^{|V|\times|V|}$ into two independent matrices $W \in \mathbb{R}^{k\times|V|}$ and $H \in \mathbb{R}^{k\times|V|}$, which satisfies $M \approx W^{\mathrm{T}}H$. Therefore, the objective function of DeepWalk algorithm based on matrix factorization is

$$\min_{W,H} \sum_{(i,j)\in\Omega} (M_{ij} - (W^{\mathrm{T}}H)_{ij})^2 + \frac{\lambda}{2}(\|W\|_{\mathrm{F}}^2 + \|H\|_{\mathrm{F}}^2) \tag{8}$$

In Eq. (8), $\|\cdot\|$ is the Frobenius norm, $\lambda/2$ weights the trade-off between $\|W\|_{\mathrm{F}}^2$ and $\|H\|_{\mathrm{F}}^2$. The minimization operation of $\|W\|_{\mathrm{F}}^2$ and $\|H\|_{\mathrm{F}}^2$ adds the low-rank constraint for matrices W and H.

Figure 2 shows the procedure of modeling network vertex relations. In practical applications, we can use commonly used matrix factorization algorithm to directly factorize matrix M, such as SVD algorithm.

TFNR algorithm adopts the IMC method applied

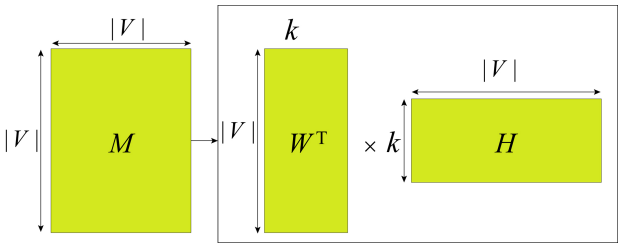


Fig. 2 Matrix factorization form of DeepWalk.

by TADW algorithm[25] to ensemble structure features, future link probabilities between vertices without edges, and text features into network representation vectors.

The objective function of IMC is as follows:

$$\min_{W,H} \sum_{(i,j)\in\Omega} (M_{ij} - (X^{\mathrm{T}}W^{\mathrm{T}}HY)_{ij})^2 + \frac{\lambda}{2}(\|W\|_{\mathrm{F}}^2 + \|H\|_{\mathrm{F}}^2) \tag{9}$$

where matrices $X \in \mathbb{R}^{p\times m}$ and $Y \in \mathbb{R}^{q\times n}$ are adopted to factorize the network structure feature matrix M. IMC aims to find matrices $W \in \mathbb{R}^{k\times p}$ and $H \in \mathbb{R}^{k\times q}$ to meet the factorization condition $M \approx X^{\mathrm{T}}W^{\mathrm{T}}HY$.

However, TFNR algorithm sets matrix X to identity matrix E, thus, the objective function of TFNR is as follows.

$$\min_{W,H} ||M - E^{\mathrm{T}}W^{\mathrm{T}}HY\|_{\mathrm{F}}^2 + \frac{\lambda}{2}(\|W\|_{\mathrm{F}}^2 + \|H\|_{\mathrm{F}}^2) \tag{10}$$

In order to intuitively understand Eq. (10), we give a detailed factorization diagram of Eq. (10) in Fig. 3.

As shown in Fig. 3, there exist three matrices $M \in \mathbb{R}^{|V|\times|V|}$, $E \in \mathbb{R}^{|V|\times|V|}$, and $Y \in \mathbb{R}^{s\times|V|}$. TFNR algorithm aims at finding matrices $W \in \mathbb{R}^{k\times|V|}$ and $H \in \mathbb{R}^{k\times s}$ to satisfy the factorization condition $M \approx E^{\mathrm{T}}W^{\mathrm{T}}HY$, k is the column size of matrix W.

In Fig. 3, we set matrix Y as an auxiliary feature matrix to factorize the network structure feature matrix M, namely, we set the future link probabilities between vertices without edges and text features as matrix Y. Importantly, we also try other feature integration methods, for example, we replace the identity matrix E with the future link probability matrix, and replace the matrix Y with the text feature matrix, but the experimental results show that this kind of feature integration approach gets the worst performance of network vertex classification compared with DeepWalk. Therefore, we first integrate the text feature matrix and the future link probability matrix based on the matrix multiplication form, i.e., $M^{\text{probability}}_{|V|\times|V|} \times T_{|V|\times d}$, then we reduce the dimension of matrix $M^{\text{probability}}_{|V|\times|V|} \times T_{|V|\times d}$ using SVD algorithm, where $M^{\text{probability}}_{|V|\times|V|} \times T_{|V|\times d} \approx U \times S \times V$. Finally, we use the matrix $U \times \sqrt{S}$ to replace parameter Y in IMC algorithm. Note that matrix $M^{\text{probability}}$ has a size of $|V| \times |V|$, matrix T has a size of $|V| \times d$, d is the column size of text feature matrix. Generally, the size of d is the same with the size of k. Consequently, we denote $W^{\mathrm{T}} \oplus Y^{\mathrm{T}}H^{\mathrm{T}}$ as the final network representation vectors, which has a column size of $2k$.

3 Experiment and Analysis

In our experiment, we conduct vertex classification tasks on three real-world datasets to evaluate the presented model. Meanwhile, we also visualize our learnt representations of three networks to verify whether TFNR is qualified to learn discriminative representations. We also show the results of the algorithm parameter sensitivity.

3.1 Dataset setup

Network vertex classification, visualization, link prediction, and other tasks are generally used to measure the performance of network representation learning algorithm. The case study is also used to compare and analyze the properties of network representation vectors. In addition, the performance of network representation learning algorithm is mainly measured on real-world citation network datasets and social network datasets. In order to find the best parameter combination, parameter sensitivity analysis of network representation learning algorithm is also done on different datasets.

It can be found from Table 1 that the average clustering coefficients of Citeseer, DataBase systems and Logic Programming (DBLP), and Cora datasets are almost the same. According to the average path length, DBLP is a dense network dataset compared with Cora and Citeseer, and Citeseer is a sparse network dataset. Cora is a dense dataset compared with Citeseer, but it

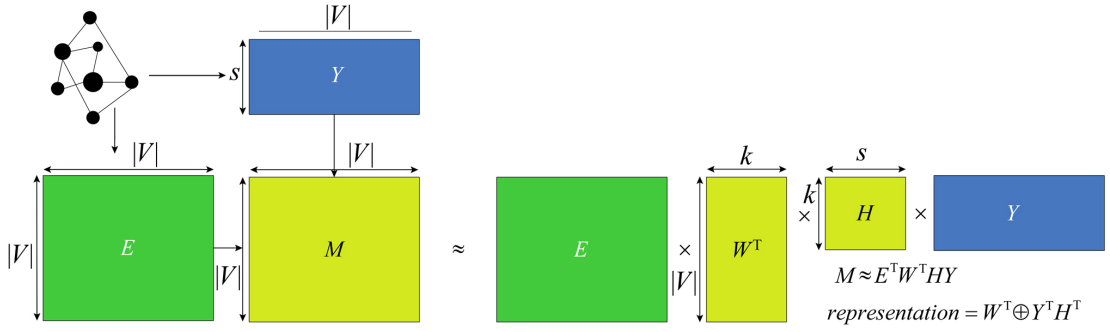


Fig. 3 Framework of modeling network vertex relations by TFNR algorithm.

Table 1 Dataset description.

Dataset	Number of nodes	Number of edges	Number of categories	Average degree	Network diameter	Average path length	Graph density	Average clustering coefficient
Citeseer	3312	4732	6	2.857	28	9.036	0.001	0.257
DBLP	3119	39 516	4	25.339	14	4.199	0.008	0.259
Cora	2708	5429	7	4.01	19	6.31	0.001	0.293

is also a sparse network dataset compared with DBLP. Thus, network representation learning performance of TFNR and different baseline algorithms on three network datasets can be measured by using network datasets with different sparsity.

3.2 Baseline algorithms

DeepWalk. DeepWalk[5] is commonly used to do a performance comparison with the proposed improved algorithms of network representation learning. We use the Skip-Gram model and Hierarchical Softmax to construct DeepWalk algorithm. We set window size as 5, random walk length as 80.

LINE. LINE[7] only considers the first-order similarity and second-order similarity between vertices, so LINE is faster than DeepWalk algorithm in modeling the relationship between network vertices. However, experimental results show that LINE can improve the training speed and take the loss of the accuracy of NRL. Here, we use the 2nd LINE model to learn the representation vectors of different networks.

MFDW. DeepWalk algorithm aims at factorizing the network structure feature matrix $M = (P + P^2)/2$. Thus, we factorize the matrix M to get the vertex representations.

MMDW. MMDW first factorizes the network structure feature matrix $M = (P + P^2)/2$ by SVD, and then MMDW algorithm uses matrix W as the vertex representations. Finally, MMDW introduces max-margin theory to optimize the learnt vertex representation vectors.

TEXT. We reduce the dimension size of text feature matrix T to 200 as the vertex representation vectors.

TADW. TADW factorize the network structure feature matrix M based on the text features. TADW also adopts the same factorization algorithm with TFNR.

3.3 Classifiers and experiment setup

In Section 3.2, we introduce various network representation learning algorithms used in this paper. In this section, we will talk about the parameter settings of each algorithm. For each network representation learning algorithm, we set the vector dimension of its network representation learning to 200. TADW algorithm and the TFNR algorithm proposed in this paper adopt the same text features. In the network vertex classification experiment, we set the proportion of training set from 10% to 90%, and we provide the network vertex classification accuracies of TFNR algorithm when λ is 0.1, 0.4, or 0.7. In the visualization and case study, we set λ to 0.7. We repeat our experiment 10 times, and then take the average accuracy as the final result. Finally, we use LIBLINEAR[34] as the classifier for network vertex classification tasks.

3.4 Experimental results and analysis

In order to evaluate the future link probabilities between vertices without edges, we need to decide how to weight the probability and how to choose the link prediction algorithm. Consequently, we adopt the 21 link prediction algorithms introduced in Ref. [35] to evaluate the prediction performance on Citeseer, DBLP, and Cora datasets. We set the proportion of training set as 0.7, 0.8, or 0.9, and measure the performance of each link prediction algorithm with Area Under the Curve (AUC). The detailed link prediction results are shown in Table 2.

It can be observed from Table 2 that MFI algorithm achieves the best prediction performance on Citeseer, DBLP, and Cora datasets. Therefore, TFNR algorithm uses MFI algorithm to calculate the future link probabilities between vertices without edges, and the specific calculation approach can be referred to Eq. (3).

TFNR algorithm measures its network representation learning performance through the network vertex classification tasks. Therefore, we conduct the network vertex classification tasks on the Citeseer, DBLP, and Cora datasets to measure the performance of various NRL algorithms. The detailed results are shown in Tables 3, 4, and 5.

Based on the results of Tables 3, 4, and 5, we have the following conclusions:

• DeepWalk algorithm is the most classical network representation learning algorithm and it is also the representative algorithm of network representation

Table 2 AUC on Citeseer, DBLP, and Cora datasets.

Algorithm	Train ratio	CN	Salton	Jaccard	HPI	HDI	LHN-I	AA	RA	PA	LP	Katz
Citeseer	0.7	68.13	66.32	66.51	66.29	66.03	66.47	66.37	66.37	78.98	81.06	96.89
	0.8	72.08	72.73	72.25	72.18	72.52	72.93	72.22	72.12	79.06	86.83	97.98
	0.9	74.67	74.44	74.33	74.42	74.17	74.46	74.33	74.63	79.53	88.45	97.19
DBLP	0.7	85.49	86.00	85.92	85.61	85.72	85.80	86.00	86.56	76.39	92.96	93.45
	0.8	88.40	87.92	88.26	88.95	88.31	87.87	88.22	88.50	77.13	93.65	94.18
	0.9	90.68	90.74	90.98	90.77	90.84	89.95	90.95	90.81	77.54	94.94	94.83
Cora	0.7	69.50	69.38	69.25	69.38	69.52	69.19	69.35	69.47	71.50	80.12	90.89
	0.8	72.38	72.13	72.00	72.44	72.53	72.16	72.66	72.47	71.91	82.97	92.14
	0.9	78.19	77.89	77.09	77.93	76.67	77.30	77.60	77.97	71.50	87.90	94.44

Algorithm	Train ratio	LHNII	LNBAA	LNBCN	LNBRA	ACT	Cos+	LRW	SRW	MFI	TSCN
Citeseer	0.7	95.76	66.37	66.70	66.05	75.88	88.57	87.21	86.34	96.68	84.26
	0.8	96.85	72.64	72.27	72.23	75.59	89.38	90.13	90.05	98.00	85.68
	0.9	96.20	74.52	74.25	74.27	73.79	88.49	91.25	90.47	97.80	86.27
DBLP	0.7	90.86	86.07	85.60	85.86	79.00	91.53	92.75	90.50	95.13	91.25
	0.8	91.80	88.42	88.47	88.91	80.07	93.47	93.35	92.25	96.00	91.03
	0.9	92.80	91.12	90.80	91.23	80.84	95.08	94.09	94.06	97.07	92.34
Cora	0.7	89.41	69.42	69.50	69.32	74.11	90.25	88.48	88.40	93.13	88.35
	0.8	90.37	72.50	72.19	72.84	73.67	90.98	90.58	90.50	94.25	90.64
	0.9	93.64	78.01	77.79	77.74	74.00	93.22	93.63	93.62	95.60	93.98

Table 3 Accuracy of vertex classification on Citeseer.

(%)

Labeled nodes (%)	DW	MFDW	LINE	MMDW	TEXT	TADW	TFNR ($\lambda = 0.1$)	TFNR ($\lambda = 0.4$)	TFNR ($\lambda = 0.7$)
10	48.31	49.78	39.82	55.49	53.37	68.19	70.68	70.40	70.91
20	50.36	54.80	46.83	60.70	62.18	70.03	73.24	73.47	73.34
30	51.33	56.66	49.02	63.66	68.24	71.67	73.89	74.37	74.49
40	52.31	56.75	50.65	65.27	69.71	72.45	74.84	75.22	74.49
50	52.85	57.90	53.77	66.02	71.52	73.76	75.14	74.95	75.76
60	53.33	58.32	54.20	69.14	72.27	74.06	74.90	75.83	75.56
70	52.98	58.60	53.87	69.34	72.75	74.48	75.10	75.12	75.73
80	53.47	58.29	54.67	69.47	72.76	74.74	75.38	75.99	76.44
90	53.71	57.07	53.82	69.72	72.21	75.59	74.68	76.07	75.68

Table 4 Accuracy of vertex classification on DBLP.

(%)

Labeled nodes (%)	DW	MFDW	LINE	MMDW	TEXT	TADW	TFNR ($\lambda = 0.1$)	TFNR ($\lambda = 0.4$)	TFNR ($\lambda = 0.7$)
10	81.84	75.06	79.13	79.70	59.83	81.09	83.51	84.20	83.71
20	82.41	80.82	79.81	82.05	67.34	82.43	84.95	85.23	85.02
30	83.25	83.00	80.41	84.23	70.58	83.42	85.47	85.53	85.55
40	83.74	83.96	81.22	84.84	72.60	83.74	85.81	85.92	86.02
50	83.98	84.70	82.95	83.45	73.52	84.16	86.31	86.02	86.10
60	84.24	84.94	83.39	85.42	74.50	84.40	86.38	86.36	86.20
70	84.55	85.72	83.04	84.96	75.01	84.91	86.28	86.74	86.36
80	84.26	84.62	84.74	85.78	74.57	85.26	86.39	85.96	86.45
90	83.53	85.11	83.85	84.49	74.05	86.05	86.82	85.43	86.88

learning based on neural network. MFDW algorithm is the matrix factorization form of DeepWalk algorithm, and DeepWalk adopts the random walk strategy and neural network to avoid the procedure of directly constructing the network structure feature matrix. However, the network structure feature matrix constructed by MFDW can embed the first-order and second-order relations between network vertices into the feature matrix. Experimental results show that the network vertex classification performance of MFDW algorithm is better than that of DeepWalk algorithm on relatively sparse network datasets, such as Citeseer

Table 5 Accuracy of vertex classification on Cora.

(%)

Labeled nodes (%)	DW	MFDW	LINE	MMDW	TEXT	TADW	TFNR ($\lambda = 0.1$)	TFNR ($\lambda = 0.4$)	TFNR ($\lambda = 0.7$)
10	73.29	66.38	65.13	73.61	57.05	80.69	82.61	83.83	84.40
20	75.46	75.52	70.17	79.99	62.39	81.70	85.70	86.07	85.64
30	76.19	78.78	72.20	80.43	66.25	84.47	86.85	86.59	86.80
40	77.49	80.54	72.92	81.92	69.64	85.94	86.85	87.00	87.03
50	77.89	82.09	73.45	83.76	72.99	86.35	87.45	87.25	86.85
60	77.83	81.93	75.67	84.97	74.38	86.27	87.77	87.46	87.18
70	78.86	82.62	75.25	86.39	75.07	85.97	87.57	87.97	87.64
80	79.05	81.57	76.78	86.70	75.79	87.36	87.38	87.54	87.97
90	78.62	83.81	79.34	87.45	75.70	87.70	87.96	87.04	87.44

and Cora. But MFDW and DeepWalk algorithm achieve almost the same network vertex classification performance on dense DBLP dataset.

• LINE can improve the training speed of the NRL algorithms through the loss of learning accuracy, but LINE is very suitable for large-scale network learning tasks, for example, LINE obtains slightly inferior to DeepWalk algorithm on the dense DBLP dataset, but its training speed is much faster than DeepWalk. MMDW is also a network representation learning algorithm based on matrix factorization, which adopts the node labels to optimize the network representation vectors. Therefore, the network representation learning performance of MMDW algorithm is also better than DeepWalk, LINE, MFDW, and other algorithms. Specifically, MMDW further optimizes the network representation vectors trained by MFDW algorithm. The experimental results show that these optimizations are feasible and effective.

• On DBLP and Cora datasets, the network vertex classification performance of TEXT is worse than that of DeepWalk and MFDW. However, if the target factorization matrices of MFDW and TEXT are integrated by IMC algorithm, the integrated algorithm is called as TADW algorithm, and the network vertex classification performance of TADW algorithm is better than that of MFDW and TEXT. On Citeseer dataset, the network vertex classification performance of TEXT is better than that of MFDW, and the network vertex classification performance of TADW algorithm is superior to that of MFDW and TEXT after integrating structure feature matrix and text feature matrix. These results show that the integrated features can fully reflect the features of the network structural properties if the different properties of the network feature matrices are integrated. In addition, the network representation learning algorithms based on multi-view feature integration can generate excellent network vertex classification performance, which is superior to that of single network learning algorithm.

• Inspired by TADW algorithm, TFNR algorithm proposed in this paper tries multiple feature integration methods. Finally, we find the best feature integration method for network representation learning tasks, which integrates the text features of the network vertices and the future link probabilities between vertices without edges by matrix multiplication form. TFNR replaces the text feature matrix with the concatenated feature matrix based on the framework of TADW algorithm. Experimental results show that TFNR algorithm achieves excellent network representation learning performance under the three different parameter settings. And TFNR algorithm introduces the future link probabilities between network vertices without edges based on TADW algorithm, consequently, its performance is better than that of TADW algorithm on the classification tasks. These results show that the classification performance of the network representation learning algorithm can be effectively and stably improved by introducing the link probabilities between vertices without edges.

These observations demonstrate that TFNR can learn high quality of network representations. Moreover, the classification accuracy of TFNR is also competitive though we do not perform specific optimization for the classification tasks.

3.5 Parameter sensitivity

In the previous sections, we evaluate and analyze the network vertex classification performance of TFNR algorithm on Citeseer, DBLP, and Cora datasets. In order to make a fair comparison with baseline algorithms, we uniformly set the network representation vector length k and the trade-off of the low-rank constraint λ in Eq. (10), respectively. In this section, we use the network vertex classification task to analyze

the effects of different sizes of k and λ. Note that the size of the network representation vector is equivalent to the size of the column dimension of $W^{\mathrm{T}} \oplus Y^{\mathrm{T}} H^{\mathrm{T}}$. The detailed results are shown in Fig. 4.

As shown in Fig. 4, we set the size of network representation vector to 50, 100, 150, 200, and 300. The experimental results show that TFNR achieves poor network representation learning performance when the size of network representation vector is 50 on Citeseer, DBLP, and Cora datasets. When the network representation vector size is 300, TFNR achieves better network representation learning performance. These results show that the classification performance of TFNR algorithm increases with the growth of the network representation vector size.

In addition, we set the size of λ to 0.1, 0.2, 0.3, 0.5, 0.8, and 1.0. Although the value of λ varies from 0.1 to 1.0, the network vertex classification performance of TFNR algorithm almost remains stable. Therefore, λ has negligible effect on network vertex classification of TFNR algorithm.

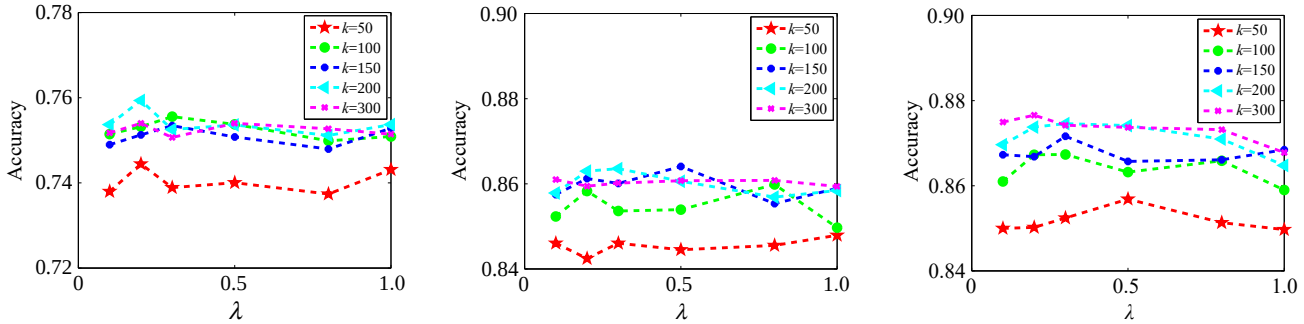
3.6 Visualization

Network representation vector visualization is another measure method of NRL, where the network representation vectors are projected to 2D visualization space. If network vertices of the same categories show a stronger internal cohesion, and network vertices of different categories show a clearer classification boundary, consequently, we suggest that the proposed network representation learning algorithm learns and generates the understandable and discriminative network representation vectors. Therefore, we visualize the learnt network representation vectors trained by DeepWalk and TFNR on Citeseer, DBLP, and Cora datasets. The detailed results are shown in Fig. 5.

As shown in Fig. 5, we randomly select 4 categories on Citeseer, DBLP, and Cora datasets, and then we randomly select 200 presentation vectors from the selected 4 categories for visualization. We use t-SNE algorithm to visualize the learnt network representation vectors. The experimental results show that the network representation vectors trained by DeepWalk algorithm show the worst visualization results on the Citeseer dataset, and show good cohesion and obvious classification boundaries on DBLP and Cora datasets. The visualization results of network representation vectors obtained by TFNR algorithm proposed in this paper is obviously better than that of DeepWalk algorithm on Citeseer. On Cora and DBLP datasets, the visualization results of TFNR algorithm and DeepWalk algorithm are almost the same. Therefore, there exists little difference about the visualization results between different algorithms on the dense network datasets, while TFNR algorithm proposed in this paper can generate better visualization results on the sparse network datasets, namely, TFNR can learn the discriminative network representation vectors on the sparse networks.

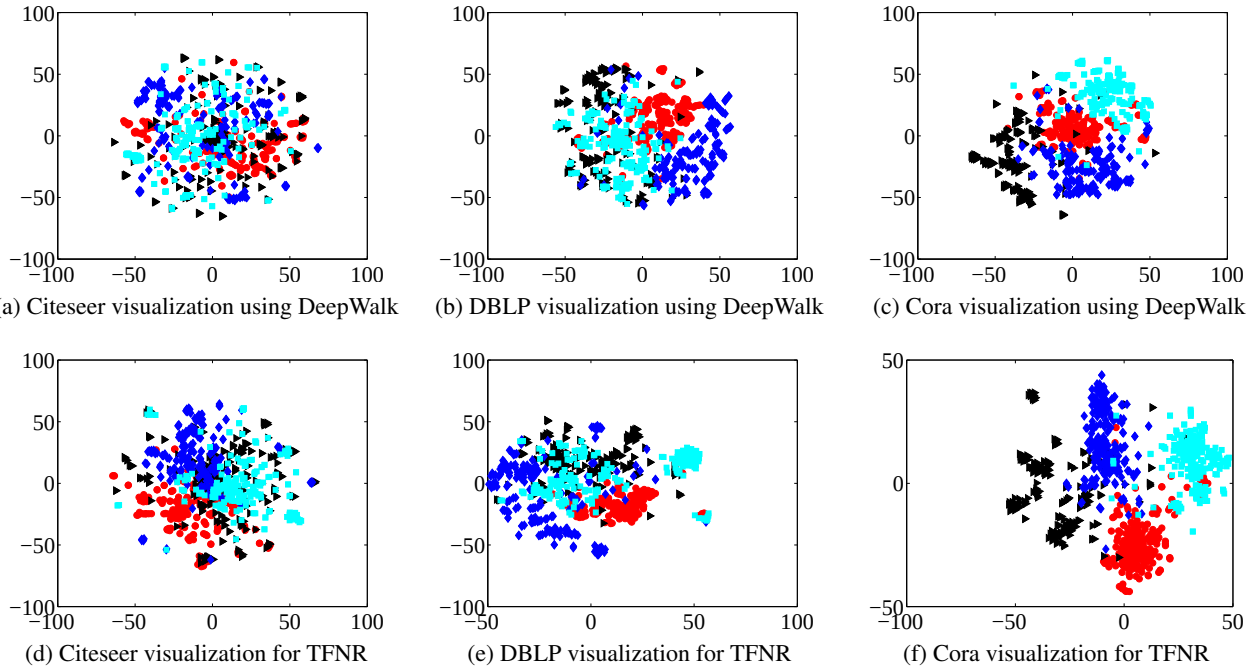
3.7 Case study

In the above sections, we verify the performance of TFNR by various tasks, such as network vertex classification and network visualization, and we also discuss the performance impacts for network vertex classification of TFNR based on network representation vector size and λ in Eq. (10). In this section, we will analyze the properties of the network representation vectors trained by the TADW and TFNR algorithms. Therefore, we first set the network title of a target vertex as "Maximum margin planning", then we adopt cosine method to calculate the five most relevant vertices of the target vertex. This experiment is a case study on the DBLP citation network. Therefore, we analyze the properties of the learnt network representation vectors by showing paper titles of the most relevant vertices.



(a) Citeseer dataset (b) DBLP dataset (c) Cora dataset

Fig. 4 λ and k sensitivity analysis on Citeseer, DBLP, and Cora datesets.



(a) Citeseer visualization using DeepWalk (b) DBLP visualization using DeepWalk (c) Cora visualization using DeepWalk

(d) Citeseer visualization for TFNR (e) DBLP visualization for TFNR (f) Cora visualization for TFNR

Fig. 5 2D visualization on Citeseer, DBLP, and Cora datasets. The abscissa and ordinate values are the two-dimensional coordinate values of the t-SNE algorithm after dimensionality reduction, in which different colors represent different types of nodes.

The specific results are shown in Table 6.

We first analyze the papers that have reference relationships with "Maximum margin planning", namely, we analyze the vertices that have link relationships with this target vertex on the DBLP network. By checking the references of the paper "Maximum margin planning", we find that the paper "Maximum margin planning" cites the papers "Solving large scale linear prediction problems using stochastic gradient descent algorithms" and "Apprenticeship learning via inverse reinforcement learning" amongst the five most relevant papers in Table 6. The paper "Learning for control from multiple tunings" cites the paper "Maximum margin planning". The papers "Robot learning from demonstration", "Algorithms for inverse reinforcement learning", and "Policy invariance under reward theory and application to reward shaping" have no any link relationships or word co-occurrence of text feature with the paper "Maximum margin planning", but it is the most relevant paper of the paper "Maximum margin planning". Therefore, we consider that one or two of these three papers and "Maximum margin planning" will be cited by a new paper in the future. Therefore, the introduction of the future link probability between vertices without edges in TFNR algorithm would make the learnt network representation vectors reflect the future evolution structures of the networks.

Table 6 Five nearest neighbors generated by DeepWalk and TFNR.

Title	Similarity	Label	Algorithm
1. Maximum margin clustering made practical	0.7063	Artificial intelligent	TADW
2. Laplace maximum margin markov networks	0.7021	Artificial intelligent	TADW
3. Fast maximum margin matrix factorization for collaborative prediction	0.6860	Artificial intelligent	TADW
4. Efficient multiclass maximum margin clustering	0.6688	Artificial intelligent	TADW
5. The relaxed online maximum margin algorithm	0.6465	Artificial intelligent	TADW
1. Robot learning from demonstration	0.6053	Artificial intelligent	TFNR
2. Apprenticeship learning via inverse reinforcement learning	0.5856	Artificial intelligent	TFNR
3. Algorithms for inverse reinforcement learning	0.5821	Artificial intelligent	TFNR
4. Learning for control from multiple demonstrations	0.5770	Artificial intelligent	TFNR
5. Policy invariance under reward transformations theory and application to reward shaping	0.5721	Artificial intelligent	TFNR

4 Conclusion

In order to introduce the future link probabilities between vertices without edges as well as text features into the network representation learning framework, we propose a novel network learning algorithm—TFNR. TFNR algorithm tries a wide variety of feature integration methods between the future link probabilities and text features, eventually, we find a best feature integration method. To embed the features of different properties into NRL framework, TFNR algorithm introduces the inductive matrix completion algorithm. Experimental results show that TFNR algorithm proposed in this paper can show excellent network vertex classification performance on Citeseer, DBLP, and Cora datasets. Through the comparison analysis of TFNR and TADW, it can be found that introducing the future link probabilities between vertices without edges can greatly improve the performance of network representation learning. In addition, the visualization experiment results show that the network representation vectors obtained by TFNR algorithm can make vertices of different categories show clearer classification boundaries, and make vertices of the same categories show stronger internal cohesion. In conclusion, the TFNR algorithm proposed in this paper is a network representation learning algorithm with excellent performance and stability.

References

[1] G. Tsoumakas and I. Katakis, Multi-label classification: An overview, *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[2] D. Liben-Nowell and J. Kleinberg, The link-prediction problem for social networks, *J. Assoc. Inform. Sci. Technol.*, vol. 58, no. 7, pp. 1019–1031, 2007.

[3] C. C. Tu, Z. Y. Liu, and M. S. Sun, Inferring correspondences from multiple sources for microblog user tags, in *Proc. 3rd Chinese National Conf. Social Media Processing*, Beijing, China, 2014, pp. 1–12.

[4] L. G. Préz, F. Chiclana, and S. Ahmadi, A social network representation for collaborative filtering recommender systems, in *Proc. 11th Int. Conf. Intelligent Systems Design and Applications*, Cordoba, Spain, 2012, pp. 438–443.

[5] B. Perozzi, R. Al-Rfou, and S. Skien, DeepWalk: Online learning of social representations, in *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2014, pp. 701–710.

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, Distributed representations of words and phrases and their compositionality, in *Proc. 26th Int. Conf. Neural Information Processing Systems*, Lake Tahoe, NV, USA, 2013, pp. 3111–3119.

[7] J. Tang, M. Qu, M. Z. Wang, M. Zhang, J. Yan, and Q. Z. Mei, LINE: Large-scale information network embedding, in *Proc. 24th Int. Conf. World Wide Web*, Florence, Italy, 2015, pp. 1067–1077.

[8] S. S. Cao, W. Lu, and Q. K. Xu, GraRep: Learning graph representations with global structural information, in *Proc. 24th ACM Int. Conf. Information and Knowledge Management*, Melbourne, Australia, 2015, pp. 891–900.

[9] D. X. Wang, P. Cui, and W. W. Zhu, Structural deep network embedding, in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 1225–1234.

[10] A. Grover and J. Leskovec, node2vec: Scalable feature learning for networks, in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 855–864.

[11] J. Tang, M. Qu, and Q. Z. Mei, PTE: Predictive text embedding through large-scale heterogeneous text networks, in *Proc. 12th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Sydney, Australia, 2015, pp. 1165–1174.

[12] X. F. Sun, J. Guo, X. Ding, and T. Liu, A general framework for content-enhanced network representation learning, http://pdfs.semanticscholar.org/fad9/08515d149bce1fe4bad84728657b8b83009a.pdf, 2018.

[13] C. C. Tu, H. Wang, X. K. Zeng, Z. Y. Liu, and M. S. Sun, Community-enhanced network representation learning for network analysis, http://pdfs.semanticscholar.org/6199/79db74a6d5896e4f21798614e80f9ce6d107.pdf, 2017.

[14] S. R. Pan, J. Wu, X. Q. Zhu, C. Q. Zhang, and Y. Wang, Tri-party deep network representation, in *Proc. 25th Int. Joint Conf. Artificial Intelligence*, New York, NY, USA, 2016, pp. 1895–1901.

[15] A. Garcia-Duran and M. Niepert, Learning graph representations with embedding propagation, http://in.arxiv.org/abs/1710.03059v1, 2017.

[16] X. Wang, P. Cui, J. Wang, J. Pei, W. W. Zhu, and S. Q. Yang, Community preserving network embedding, in *Proc. 21st AAAI Conf. Artificial Intelligence*, San Francisco, CA, USA, 2017.

[17] D. K. Zhang, J. Yin, X. Q. Zhu, and C. Q. Zhang, User profile preserving social network embedding, in *Proc. 26th Int. Joint Conf. Artificial Intelligence*, Melbourne, Australia, 2017, pp. 3378–3384.

[18] C. Z. Li, S. Z. Wang, D. J. Yang, Z. J. Li, Y. Yang, X. M. Zhang, and J. S. Zhou, PPNE: Property preserving network embedding, in *Database Systems for Advanced Applications*, S. Candan, L. Chen, T. B. Pedersen, L. J. Chang, and W. Hua, eds. Springer, 2017, pp. 163–179.

[19] X. Huang, J. D. Li, and X. Hu, Accelerated attributed network embedding, in *Proc. 2017 SIAM Int. Conf. Data Mining*, Houston, TX, USA, 2017.

[20] Z. P. Huang and N. Mamoulis, Heterogeneous information network embedding for meta path based proximity, http://pdfs.semanticscholar.org/52a1/50d6a098ef142bece099dadaa613fddbae50.pdf, 2018.

[21] K. Tu, P. Cui, X. Wang, F. Wang, and W. W. Zhu, Structural deep embedding for hyper-networks, https://arxiv.org/pdf/1711.10146.pdf, 2018.

[22] O. Levy and Y. Goldberg, Neural word embedding as implicit matrix factorization, in *Proc. 27th Int. Conf. Neural Information Processing Systems*, Montreal, Canada, 2014, pp. 2177–2185.

[23] H. F. Yu, P. Jain, P. Kar, and I. S. Dhillon, Large-scale multi-label learning with missing labels, in *Proc. 31st Int. Conf. Machine Learning*, Beijing, China, 2014, pp. 593–601.

[24] C. Yang and Z. Y. Liu. Comprehend DeepWalk as matrix factorization, https://www.researchgate.net/publication/270454626_Comprehend_DeepWalk_as_Matrix_Factorization, 2015.

[25] C. Yang, Z. Y. Liu, D. L. Zhao, M. S. Sun, and E. Y. Chang, Network representation learning with rich text information, in *Proc. 24th Int. Conf. Artificial Intelligence*, Buenos Aires, Argentina, 2015, pp. 2111–2117.

[26] C. C. Tu, W. C. Zhang, Z. Y. Liu, and M. S. Sun, Max-margin deepwalk: Discriminative learning of network representation, in *Proc. 25th Int. Joint Conf. Artificial Intelligence*, New York, NY, USA, 2016, pp. 3889–3895.

[27] N. Natarajan and I. S. Dhillon, Inductive matrix completion for predicting gene-disease associations, *Bioinformatics*, vol. 30, no. 12, pp. i60–i68, 2014.

[28] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, Support vector machines, *IEEE Intell. Syst. Their Appl.*, vol. 13, no. 4, pp. 18–28, 1998.

[29] J. Zhu, A. Ahmed, and E. P. Xing, MedLDA: Maximum margin supervised topic models, in *Proc. 26th Int. Conf. Machine Learning*, Montreal, Canada, 2009, pp. 1257–1264.

[30] S. Aouay, S. Jamoussi, and F. Gargouri, Feature based link prediction, in *Proc. 11th IEEE/ACS Int. Conf. Computer Systems and Applications*, Doha, Qatar, 2014, pp. 523–527.

[31] E. M. Dong, J. P. Li, and Z. Xie, Link prediction via convex nonnegative matrix factorization on multiscale blocks, *J. Appl. Math.*, vol. 2014, pp. 1–9, 2014.

[32] D. Li, Z. M. Xu, S. Li, and X. Sun, Link prediction in social networks based on hypergraph, in *Proc. 22nd Int. Conf. World Wide Web*, Rio de Janeiro, Brazil, 2013, pp. 41–42.

[33] A. Farasat, A. Nikolaev, S. N. Srihari, R. H. Blair, Probabilistic graphical models in modern social network analysis, *Social Network Analysis and Mining*, vol. 5, no. 1, p. 62, 2015.

[34] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin, LIBLINEAR: A library for large linear classification, *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, 2008.

[35] L. Y. Lv and T. Zhou, *Link Prediction*. Beijing, China: Higher Education Press, 2013.

Zhonglin Ye received BS degree from Sichuan University in 2012. He received the MS degree from Southwest Jiaotong University in 2016. Currently, he is working in Qinghai Normal University. His research interests include data mining, knowledge discovery, and network representation learning.

Ke Zhang received the MS degree from Qinghai Normal University in 2014. Currently, he is currently pursuing the PhD degree in Qinghai Normal University. His research interests include hypergraph theory and complex network.

Zhaoyang Wang received the MS degree from Qinghai Normal University in 2017. She is currently a master student in Qinghai Normal University. Her research interests include reliability of complex networks, social media mining, and graph theory.

Yu Zhu received the MS degree from Chang'an University in 2014. He is currently pursuing the PhD degree in Qinghai Normal University. His research interests include data mining of complex network.

Haixing Zhao received the Doctor of Engineering degree from the School of Computer Science, Northwestern Polytechnical University, China, 2004. He also received the Doctor of Science degree from University of Twente, Holland. He is a professor and part-time professor at Qinghai Normal University and Shaanxi Normal University, respectively. He is a director of Changjiang Scholars and Innovative Research Team in University, he is also the syndic of Operations Research Society, Combinatorics and Graph Theory Society, in China. His research interests include complex network, semantic network and machine translation, hypergraph theory and database, and network reliability.