World Wide Web (2020) 23:2699–2714
https://doi.org/10.1007/s11280-020-00799-7

Multi-view network embedding with node similarity ensemble

Weiwei Yuan[1,2] · Kangya He[1] · Chenyang Shi[1] · Donghai Guan[1] · Yuan Tian[3] · Abdullah Al-Dhelaan[4] · Mohammed Al-Dhelaan[4]

Received: 18 March 2019 / Revised: 2 November 2019 / Accepted: 6 February 2020 /
Published online: 12 March 2020

Abstract
Node similarity is utilized as the most popular guidance for network embedding: nodes more similar in a network should still be more similar when mapping node information from a high-dimensional vector space to a low-dimensional vector space. Most existing methods preserve a single node similarity in the network embedding, which can merely preserve one-side network structural information. Though some works try to utilize several node similarities to preserve more network information, they fail to consider the interrelationships between the latent spaces preserving different node similarities. This causes both network information insufficiency and network information redundancy. To solve the problems of existing works, we propose a novel multi-view network embedding model with node similarity ensemble. Node similarities are first selected to maximize the represented network information while minimizing the information redundancy. For each combination of the selected node similarities, a latent space is generated as a view of the network. A Canonical Correlation Analysis based approach is then used to extract the common structure of the latent spaces alignment, and a neural network based approach is used to extract the view-specific latent structure by measuring the asymmetric KL divergence of nodes' Gaussian distribution. The common structure and the view-specific structure of multiple views are merged to perverse the overall network information. Experiments held on the real-world networks verified the superiority of the proposed method to existing works.

Keywords Node similarity · Multi-view learning · Network embedding

This article belongs to the Topical Collection: *Special Issue on Application-Driven Knowledge Acquisition*
Guest Editors: Xue Li, Sen Wang, and Bohan Li

✉ Donghai Guan
dhguan@nuaa.edu.cn

Extended author information available on the last page of the article

1 Introduction

Learning vector representations for nodes in large networks has been proved extremely useful in various complex network analysis tasks [7, 9]. The main idea of network representation learning is to use dimensionality techniques to map the high-dimensional information about a node's neighbors or other inherent properties into a low dimensional space [7, 9, 12, 22]. Since machine learning techniques cannot be implemented on the network directly, the learnednode representation is essential to machine learning based graph mining tasks such as node classification, link prediction or community detection [5, 7, 9].

Real world networks can be classified into three categories based on the structure: naive networks, attributes networks and multi-layer networks. The naive network only has the basic elements of a network, namely nodes and edges [7, 18]. The attributed network consists of nodes and edges along with node attributes and edge attributes [7, 12, 21]. The multi-layer network, also called heterogeneous network, is the most complex one [22]. The nodes and edges of a multi-layer network are of different types, each type of node defines a layer, and the interactions between nodes can be either intra- or inter-layers [5, 19, 22]. Although the embedding of attributed and multi-layer networks have received increasing interests, it is not always possible to obtain additional information of network in the real applications. Thus embedding naive networks is a more general problem and this work focuses on the network embedding of naive networks.

Measuring the similarity of nodes is the most straightforward method to provide evidence for network embedding [15]. Nodes which are more similar in a network should still be more similar when mapping the network from a high-dimensional space to alow-dimensional space [16]. However, different similarity measurements have their own strengths and weaknesses [15]. For example, common neighbor based methods can easily capture the local topology similarity, but it would do nothing for the nodes who are similar with each other but have few common neighbors in the network. On the other hand, some global similarity measurements have advantages in measuring the similarity in global structure, but the precision of local similarity calculation cannot be guaranteed.

The majority of existing network embedding models merely use one kind of node similarity measurements. For example, LINE [23] captures the 1st and 2nd order relationship between nodes, but it fails to reflect 3rd or higher order relationships. Random walk based methods such as DeepWalk [17] and Node2vec [8] get the similarity information by the co-occurrence probability of nodes. Theoretically, random walk [18] methods can only get partial structural similarities. What's worse, due to the link sparseness problem existing extensively in the networks, it is difficult get global similarities in the whole graph level.

Since no single similarity measurement can capture all the structural information of network, it is natural to utilize multiple similarity measurements and combine them together in the embedding procedure. Though some works [16, 23] try to utilize several node similarities to preserve more network information, they fail to consider the interrelationships between the latent spaces preserving different node similarities. This causes both network information insufficiency and network information redundancy.

To solve the problems of existing works, we propose a novel multi-view network embedding model to capture the structural information derived from different node similarity measurements. Latent spaces are firstly generated to preserve different node similarity sets. These latent spaces are regarded as the views for multi-view network embedding. Multiple views are fused to generate an overall latent space to represent the network structural

information. The overall latent space merges the common representation of multiple views and view-specific representation of each view, in which common representation is learned via a Canonical Correlation Analysis [1, 11] based method and view-specific representation [2] is learned via a neural network based method. Experiments held on real-world datasets with node classification task verify that: the proposed model contributes to better node classification performances compared with network embedding models preserving single node similarity or simple combination of node similarities. In addition, since the proposed model preserves node similarity ensemble, it can describe the overall network structure with limited number of training samples. This contributes to its superior performances with very low ratio of training sets.

2 Problem formulation

Node similarity is the most popular measure to represent the network structural information [15, 25]:

> **Definition 1 (Node Similarity)** For a given network $G = (V, E)$, where V is the node set and E is the edge set. The node similarity function maps the node pair into a real space, which is denoted as $S: V \times V \rightarrow \mathbb{R}$. $S(v_i, v_j)$ denotes the node similarity between node v_i and v_j.

Network embedding maps the high-dimensional network information into a low-dimensional latent space [7, 9]. Node similarity is always used to guide the network embedding [26], i.e., node similarity information should be preserved in network embedding. Network embedding preserves a single set of node similarities is defined as single-view network embedding:

> **Definition 2** (**Single-view network embedding**) For a network $G = (V, E)$, using m similarity measures $\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m\}$, single-view network embedding maps each vertex $v \in V$ into a low dimensional vector space $\mathbb{R}^d$ by preserving a non-empty subset of $\mathcal{S}$. The mapping function ofthe single-view network embedding is represented as $f: V \rightarrow \mathbb{R}^d$.

Each node similarity represents the network structural information from one-side point of view. None of them can describe the whole network. For example, common neighbor based node similarities represent the local information of network, measuring relationships with nodes who are directly connected to the active node or the nodes who are within 2 hops away from the active node [10, 15]. They cannot measure global network structural information. Random walk based node similarities represent the global information of network, measuring relationships with nodes who are randomly selected in the network [10, 15, 18]. They cannot measure local network structural information [15].

As shown in Definition 2, single-view network embedding preserves a set of node similarities. However, it is still not enough to represent the overall network structural information. Instead of preserving a single set of node similarities, a combination of node similarity sets should be preserved to represent the overall network structural information.

Preserving a set of node similarities by single-view network embedding, the generated latent space is defined as a view of the latent space preserving the overall network structural

information. Multiple views are fused as the low-dimensional latent representation of the overall network. This is defined as multi-view network embedding:

Definition 3 (**Multi-view Network Embedding**) Multi-view network embedding maps multiple latent vector spaces, which are generated by multiple single-view network embedding preserving different node similarity sets, into a single low dimensional latent space to represent the network. For a given network $G = (V, E)$, the latent space candidates which are used for multi-view network embedding are represented as $\mathcal{M} = \{f_1(V), f_2(V), \ldots, f_n(V)\}$, where $f_i(V)$, $i = 1, 2, \ldots, n$, is the latent space generated by single-view network embedding. Multi-view network embedding is a mapping $F : \mathcal{M} \rightarrow \mathbf{V} \in \mathbb{R}^d$, where $\mathbf{V}$ is the multi-view latent space of G.

3 The proposed method

The proposed model generates low-dimensional latent spaces for high-dimensional network information via multi-view network embedding. Multiple latent spaces are firstly generated by single-view network embedding, preserving various node similarity ensembles. Common representation and view-specific representations are learned from these multiple latent spaces. The learned representations are merged as the latent space of multi-view network embedding to represent the network. The details of the proposed method are as follows.

3.1 Single-view network embedding

Single-view network embedding preserves node similarities of a non-empty set of $\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m\}$, where $\mathbf{S}_i$ is the similarity matrix between node of V with the i^{th} node similarity measurement, and m is the total number of node similarities.

To preserve network information from node similarity point of view, single-view network embedding should minimize the difference between node similarities of the original network $\mathcal{S}_i$ and node similarities of the latent network space $\mathbf{S}_i^{'}$. Sigmoid function [18] is used in this work to measure the node similarity of the latent representation space:

$$s^{'}\left(v_i, v_j\right) = \frac{1}{1 + e^{-\mathbf{v}_i^{\mathbf{T}}\mathbf{v}_\mathbf{j}}} \tag{1}$$

where $\mathbf{v}_i$ and $\mathbf{v}_j$ denote the representation vector of node v_i and v_j in the representation vector space. The similarity matrix in the representation space is denoted as $\mathbf{S}'$, where $\mathbf{S}'[i, j] = s'(v_i, v_j)$.

This work focuses on the directed network, and the representation of each node preserves two kinds of similarities: the similarities with nodes in which the active node acts as the source nodes of links, and the similarities with nodes in which the active node acts as the target node of links. For a node v_i, let the source node representation $\mathbf{X}_i$ and the target node representation $\mathbf{Y}_i$ be the representation preserving these two kinds of node similarities. The similarity matrix in representation space is:

$$\mathbf{S}_i^{'} = Sigmoid\left(\mathbf{X}_i \cdot \mathbf{Y}_i^T\right) \tag{2}$$

where $\mathbf{S}_i^{'}[j, k] = 1/\left(1 + e^{-\mathbf{x}_j^{\mathbf{T}}\mathbf{y}_k}\right)$.

The similarity difference function is defined based on KL Divergence [2, 3]:

$$L = \sum_{\mathbf{S}_j \in \mathcal{S}_i} \sum_{k=1}^{|V|} \sum_{l=1}^{|V|} \mathbf{S}_j[k,l] \log \frac{\mathbf{S}_j[k,l]}{\mathbf{S}_i'[k,l]}. \tag{3}$$

Since $\log \frac{\mathbf{S}_j[k,l]}{\mathbf{S}_i[k,l]}$ may be less than 0, $\log\left(1 + \frac{\mathbf{S}_j[k,l]}{\mathbf{S}_i[k,l]}\right)$ is used to substitute $\log \frac{\mathbf{S}_j[k,l]}{\mathbf{S}_i[k,l]}$ in (3). Let $\mathbf{X}_i^*$ and $\mathbf{Y}_i^*$ be the optimal representation of $\mathbf{X}_i$ and $\mathbf{Y}_i$. $\mathbf{X}_i^*$ and $\mathbf{Y}_i^*$ can be calculated as:

$$\left(X_i^*, Y_i^*\right) = \min_{X_i^*, Y_i^*} \sum_{S_j \in \mathcal{S}_i} \sum_{k=1}^{|V|} \sum_{l=1}^{|V|} S_j[k,l] \log\left(1 + \frac{S_j[k,l]}{S_i'[k,l]}\right). \tag{4}$$

For a node v_i, its latent vector representation with single-view embedding is denoted as $f_i(V)$, in which $f_i(.)$ is the embedding mapping function by preserving the i^{th} node similarity set. $f_i(V)$ is the combination of X_i^* and Y_i^*:

$$f_i(V) = \frac{X_i^* + Y_i^*}{2}, \tag{5}$$

where X_i^* and Y_i^* are calculated by (4).

3.2 Common representation with latent space alignment

By preserving network structural information of n node similarity sets with single-view network embedding, the candidates of multi-view network embedding are achieved: $\mathcal{M} = \{f_1(V), f_2(V), \cdots, f_n(V)\}$, where $f_i(V)$, $i = 1, 2, ..., n$, is the latent space by preserving the node similarity information of the i^{th} node similarity set with single-view network embedding. To merge these n latent spaces, their common representation is firstly extracted.

Let $\mathfrak{V}$ be an underlying space which is observable by the latent spaces of $\mathcal{M}$. A mapping function $H_i : \mathfrak{V} \rightarrow \mathbb{R}^{d'}$ $(i = 1, 2, .., n)$ is used to represent the projection from $\mathfrak{V}$ to the latent spaces. For a node $v_i \in V$, let $\mathscr{V}_i \in \mathfrak{V}$ be its representation in $\mathfrak{V}$, and $f_1(v_i), f_2(v_i), ..., f_n(v_i)$ are its latent vector representations generated by single-view network embedding, $H_j(\mathscr{V}_i) = f_j(v_i)$, $j = 1, 2, .., n$ [27]. Then we can get:

$$\mathscr{V}_i = H_1^{-1}(f_1(v_i)) = ... = H_n^{-1}(f_n(v_i)). \tag{6}$$

H_j^{-1} $(j = 1, 2, .., n)$ should satisfy (6) to achieve the optimal solution. However, since 6 cannot be solved directly, motivated by the Canonical Correlation Analysis (CCA) [1, 11, 20], a series of mapping functions H_j^{-1} $(j = 1, 2, .., n)$, which maximize the correlation of two vector spaces' projections, are used as the approximate solution of (6). Therefore, (6) is relaxed to the following optimal problem:

$$\begin{aligned} &\left(H_1^{-1*}, H_2^{-1*}, ..., H_n^{-1*}\right) = \\ &\max_{H_1^{-1}, H_2^{-1}, ..., H_n^{-1}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \rho\left(H_i^{-1}(f_i(V)), H_j^{-1}\left(f_j(V)\right)\right) \end{aligned} \tag{7}$$

where $\rho\left(H_i^{-1}(f_i(V)), H_j^{-1}\left(f_j(V)\right)\right)$ is the correlation of two latent vector spaces' projections $H_i^{-1}(f_i(V))$ and $H_j^{-1}\left(f_j(V)\right)$.

To simplify the calculation of (7), linear transformation $\mathbf{W}_i \in \mathbb{R}^{d \times d'}$ is used to approximate H_i^{-1} [1], and (7) is rewritten as:

$$\left(\mathbf{W}_1^*, \mathbf{W}_2^*, ..., \mathbf{W}_n^*\right) = \max_{W_1, W_1, ..., W_n} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \rho\left(\mathbf{W}_i^T \cdot f_i(V), \mathbf{W}_j^T \cdot f_j(V)\right) \tag{8}$$

Since $\mathbf{W}_1$, $\mathbf{W}_1$, ..., $\mathbf{W}_n$ are mutually constrained, it is difficult to solve the optimization problem in (8) [20]. We therefore design a CCA based common representation learning model to fuse all candidate latent spaces. Two latent spaces of the multi-view network embedding candidates are combined to generate a new latent space, and this new latent space is added to the multi-view network embedding candidates to substitute the original two latent spaces. This process is iteratively executed until there is only one latent space left in the multi-view network embedding candidate set. This latent space is the common representation of the latent space alignment. The details of CCA based common representation learning model is given in Algorithm 1. $\mathcal{M}_0$ is the latent space candidate set that need to be transformed, and $\mathcal{M}_1$ denotes the transformed latent space set. In each iteration, two latent spaces $\mathbf{V}_1$ and $\mathbf{V}_2$ of $\mathcal{M}_0$ are fused to generate a new latent space $\mathbf{V}^*$, which is the average of the projection of $\mathbf{V}_1$ and the projection of $\mathbf{V}_2$. $\mathbf{V}^*$ is added to $\mathcal{M}_1$ for thenext iteration of latent space fusing, and this process terminates until the size of $\mathcal{M}_0$ is one. The final latent space of $\mathcal{M}_0$ is the common representation of $\mathcal{M} = \{f_1(V), f_2(V), ..., f_n(V)\}$, which is represented as $\mathbf{V}_{common} \in \mathbb{R}^{|V| \times d'}$.

3.3 View-specific representation learning

For multiple views generated by single-view network embedding, view-specific representation are extracted for view fusion. This is achieved by representing nodes' characteristics as Gaussian distributions [2]: each node's characteristic is described as a full distribution rather than a single vector, by measuring the Gaussian distributions of nodes when preserving node similarities, view-specific representation is learned.

For a network $G = (V, E)$, its generated latent spaces by single-view network embedding is $\mathcal{M} = \{f_1(V), f_2(V), ..., f_n(V)\}$. To simplify the illustration, we denote the concatenation of $\mathcal{M}$ as $\mathbf{M} = [f_1(V), f_2(V), ..., f_n(V)] \in \mathbb{R}^{|V| \times (nd)}$. In this part, we aim to find a lower-dimensional Gaussian distribution embedding $v_{specific}^i = \mathcal{N}(\mu_i, \Sigma_i)$, $\mu_i \in \mathbb{R}^{d'}$, $\Sigma_i in \mathbb{R}^{d' \times d'}$ for every node v_i [2].

For a node $v_i \in V$, we define k-hop neighborhood [25] which is the set of nodes who are k hops away from node v_i: $N_{ik} = \{v_j \in V | v_i \neq v_j, sp(v_i, v_j) = k\}$, where $sp(v_i, v_j)$ is the length of the shortest path from v_i to node v_j. Nodes in different neighborhood should satisfy [2]:

$$d\left(\mathcal{N}_i, \mathcal{N}_j\right) < d\left(\mathcal{N}_i, \mathcal{N}_{j'}\right), \forall v_i \in V, v_j \in N_{ik}, v_{j'} \in N_{ik'}, k < k', \tag{9}$$

Algorithm 1: CCA Based Common Representation Learning Model [1] different representation learned from different views $\mathcal{M} = \{f_1(V), f_2(V), \ldots, f_n(V)\}$, the dimension of common representation d'. the common embedding space $\boldsymbol{V}_{common} \in \mathbb{R}^{|V| \times d'}$

$\mathcal{M}_0 \leftarrow \mathcal{M}$
$\mathcal{M}_1 \leftarrow Empty$, temporary set for computation
$k \leftarrow 0$, temporary index number $|\mathcal{M}_0| > 1$
$\mathcal{M}_1 \leftarrow Empty$ $|\mathcal{M}_0|$ is a odd number
$k = \frac{|\mathcal{M}_0|-1}{2}$
$\mathcal{M}_1 \leftarrow \mathcal{M}_1 \cup \mathcal{M}_0[|\mathcal{M}_0|]$
$k = \frac{|\mathcal{M}_0|}{2}$ $i = 0,1,2 \ldots k$
$\boldsymbol{V}_1 \leftarrow \mathcal{M}_0[2i]$
$\boldsymbol{V}_2 \leftarrow \mathcal{M}_0[2i+1]$
$(\boldsymbol{W}_1^*, \boldsymbol{W}_2^*) = \max \rho\,(\boldsymbol{W}_1^T \cdot \boldsymbol{V}_1, \boldsymbol{W}_2^T \cdot \boldsymbol{V}_2)$
$\boldsymbol{V}^* \leftarrow \frac{(\boldsymbol{W}_1^* \cdot \boldsymbol{V}_1 + \boldsymbol{W}_2^* \cdot \boldsymbol{V}_2)}{2}$
$\mathcal{M}_1 \leftarrow \mathcal{M}_1 \cup \boldsymbol{V}^*$
$\mathcal{M}_0 \leftarrow \mathcal{M}_1$
$\boldsymbol{V}_{common} \leftarrow \mathcal{M}_0[0]$
Return $\boldsymbol{V}_{common}$

where $\mathcal{N}_i$, $\mathcal{N}_j$ and $\mathcal{N}_{j'}$ are the corresponding Gaussian distribution of node v_i, v_j and $v_{j'}$ respectively, and N_{ik}, $N_{ik'}$ are the k-hop and k'-hop neighborhoods of v_i. To solve (9), asymmetric KL divergence is used to measure the difference between two normal distributions [2, 3]. Given the latent Gaussian distribution representation of two nodes v_i and v_j we define:

$$d(\mathcal{N}_i, \mathcal{N}_j) = D_{KL}\left(\mathcal{N}_j \| \mathcal{N}_i\right) = \frac{1}{2}\left[tr\left(\Sigma_i^{-1}\Sigma_j\right) + \left(\mu_i - \mu_j\right)^T \Sigma_i^{-1}\left(\mu_i - \mu_j\right) - d' - \log\left(\frac{det(\Sigma_j)}{det(\Sigma_i)}\right)\right], \tag{10}$$

where μ_i and Σ_i are the parameters of normal distribution of $\mathcal{N}_i$, μ_j and Σ_j are the parameters of normal distribution of $\mathcal{N}_j$, d' is the dimension of Gaussian distribution $\mathcal{N}_i$ and $\mathcal{N}_j$, $tr(.)$ denotes the trace of a matrix, and $det(.)$ denotes the determinant value of a matrix.

A neural network with two hidden layer is then used to calculate the parameters of the Gaussian distribution by learning the Gaussian distribution from the concatenation $\mathbf{M} = [f_1(V), f_2(V), \cdots, f_n(V)] \in \mathbb{R}^{|V| \times (nd)}$ [2]. The first layer of the neural network is defined as:

$$h_1(\mathbf{m}_i) = Relu(\Theta_1 \cdot \mathbf{m}_i + \mathbf{b}_1), \tag{11}$$

where $\mathbf{m}_i$ is the representation of node v_i in $\mathbf{M}$, Θ_1 and $\mathbf{b}_1$ are the weight and bias of the first layer, and $Relu(.)$ is the activation function [6]. The diagonal covariance matrices are used to represent the covariance of the Gaussian distribution [2]. the parameters of node v_i's Gaussian distribution are defined as:

$$\mu_i = Relu\left(\Theta_\mu \cdot h_1(\mathbf{m}_i) + \mathbf{b}_\mu\right), \tag{12}$$

$$\Sigma_i = Relu(\Theta_\Sigma \cdot h_1(\mathbf{m}_i) + \mathbf{b}_\Sigma), \tag{13}$$

where Θ_μ and $\mathbf{b}_\mu$ are the weight and bias of the output layer calculating μ, and Θ_Σ and $\mathbf{b}_\Sigma$ are the weight and bias of the output layer calculating Σ.

To satisfy the constraints of (9) with (10), (11), (12) and (13), an objective function is defined to penalize the ranking errors according to the energy of pairs. KL divergence between a pair of nodes v_i and v_j is defined as their energy: $\mathcal{E}_{ij} = D_{KL}(\mathcal{N}_j \| \mathcal{N}_i)$ [2]. The loss function is therefore defined as:

$$\begin{aligned} L &= \sum_{v_i \in V} \sum_{k<k'} \sum_{v_j \in N_{ik}, v_{j'} \in N_{ik'}} \left(\mathcal{E}_{ij}^2 + \exp^{-\mathcal{E}_{ij'}} \right) \\ &= \sum_{\left(v_i, v_j, v_{j'}\right) \in \mathcal{T}} \left(\mathcal{E}_{ij}^2 + \exp^{-\mathcal{E}_{ij'}} \right), \end{aligned} \tag{14}$$

where $\mathcal{T} = \left\{ \left(v_i, v_j, v_{j'}\right) | sp\left(v_i, v_j\right) < sp\left(v_i, v_{j'}\right) \right\}$ is the set of all valid triplets. $\mathcal{E}_{ij}$ is the set of positive examples whose energy is lower than the set of the negative examples $\mathcal{E}_{ij'}$. For agiven target node v_i, the energy $\mathcal{E}_{ij}$ should be the lowest for nodes v_j in its 1-hop neighborhood, followed by a higher energy for nodes in its 2-hop neighborhood and so on [2].

The optimal solution of (14) is the view-specific representation for the candidate latent space of $\mathcal{M} = \{f_1(V), f_2(V), ..., f_n(V)\}$.

View-specific representation is extracted via Gaussian distribution $v^i_{specific} = \mathcal{N}(\mu_i, \Sigma_i)$. $\mu_i \in \mathbb{R}^{d'}$ is a vector, while $\Sigma_i \in \mathbb{R}^{d' \times d'}$ is a diagonal matrix. μ_i and the diagonal element of Σ_i are concatenated to a $2d'$ dimension vector. $\mathbf{v}^i_{specific} = \left[\mu_i, \Sigma_i^{'}\right]$ is used to denote the view-specific representation node v_i, where $\Sigma_i^{'}$ is the the diagonal element of Σ_i. $\mathbf{V}_{specific}$ is used to denote the view-specific embedding space of the candidate latent spaces.

3.4 Joint representation with common and view-specific representations

With the latent spaces generated by single-view network embedding, common representation are extracted with the latent space alignment, and view-specific representations are extracted to reveal the characteristics of the latent spaces. The latent vector representation of multi-view network embedding is the joint representation of the common representation and view-specific representations. The output of multi-view network embedding is $\mathbf{V} = [\mathbf{V}_{common}, \mathbf{V}_{specific}]$.

4 Model optimization

4.1 Node Similarity Ensemble Strategy

Single-view network embedding generates a single view of latent network representation by preserving a set of node similarities. Suppose there are m node similarities, i.e., $\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_m\}$, the number of possible node similarity sets is $2^m - 1$. If

latent spaces are generated for all possible node similarity sets by single-view network embedding, there are $2^m - 1$ views need to be fused for the final latent representation of the multi-view network embedding. If m is large, the computational complexity of merging these views is high. In addition, it is not necessary to generate views for all possible node similarity sets. Since some subsets belong to other subsets of S, there is great information redundancy in the $2^m - 1$ non-empty subsets of S.

To reduce the computational complexity and the information redundancy, the most valuable node similarity sets should be selected for single-view network embedding. The latent spaces preserving the most valuable node similarity sets maximize the preserved network structural information while minimize the information redundancy. These views are then merged as the latent representation of the proposed multi-view network embedding model.

Node similarity ensemble strategy is defined as follows: for node similarity set $\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_m\}$, its optimal node similarity ensemble set $\mathcal{S}_{candidate}$ should satisfy the following constraints:

(1) $\forall \mathcal{S}_i \in \mathcal{S}_{candidate}$, then $\mathcal{S}_i \subseteq \mathcal{S}$ and $\mathcal{S}_i \neq \Phi$;
(2) $\forall \mathcal{S}_i, \mathcal{S}_j \in \mathcal{S}_{candidate}$, then $\mathcal{S}_i \notin \mathcal{S}_j$ and $\mathcal{S}_j \notin \mathcal{S}_i$;
(3) $\mathcal{S}_{candidate}$ should be as large as possible.

Node similarity ensemble strategy is a k-combination problem of $\mathcal{S}$. If $k = \lfloor \frac{m}{2} \rfloor$, where $\lfloor . \rfloor$ is the rounding down function, $\mathcal{S}_{candidate}$ satisfies the above constraints. The optimal node similarity ensemble set is therefore represented as: $\mathcal{S}_{candidate} = \left\{ \mathcal{S}_i | \mathcal{S}_i \subseteq \mathcal{S}, |\mathcal{S}_i| = \lfloor \frac{m}{2} \rfloor \right\}$

4.2 Triplet Sampling Strategy

For a large scale network G, it is hard to compute the complete loss in (14) since the amount of triplets in $\mathcal{T}$ is extremely large. A naive approach to address this problem is to sample triplets from $\mathcal{T}$ uniformly [2], i.e. replace $\sum_{\left(v_i, v_j, v_{j'}\right) \in \mathcal{T}}$ with $\mathbb{E}_{\left(v_i, v_j, v_{j'}\right) \sim \mathcal{T}}$ in (14). However, using this naive sampling approach, low-degree nodes are less likely to be sampled in the triplets compared with high-degree nodes. The representations of low-degree nodes are therefore hard to be updated. Since the degree distribution of complex networks follows the power-law [24], majority of nodes have low degree. The naive approach is not effective for most nodes of the network.

Triplet sampling strategy used in this work is defined as follows [2]: for a node v_i, one node is randomly sampled from each of v_i's neighborhoods (N_{i1}, N_{i2}, etc.), and triplets are then generated from these randomly sampled nodes. The constraints of v_i is $\mathcal{E}_{i1} < \mathcal{E}_{i2}$, $\mathcal{E}_{i1} < \mathcal{E}_{i3}$, $\cdots$, $\mathcal{E}_{i1} < \mathcal{E}_{iK}$, $\mathcal{E}_{i2} < \mathcal{E}_{i3}$, $\mathcal{E}_{i3} < \mathcal{E}_{i4}$, $\cdots$, $\mathcal{E}_{i2} < \mathcal{E}_{iK}$, $\cdots$, $\mathcal{E}_{iK-1} < \mathcal{E}_{iK}$, where K is the radius of the network.

Using triplet sampling strategy, the parameters of view-specific representation learning can be optimized. Parameters (Θ_1, $\mathbf{b}_1$,Θ_μ, $\mathbf{b}$, Θ_Σ, $\mathbf{b}$) of neural network model are optimized such that the loss L (in (14)) is minimized and the sampled triplets set $\mathcal{T}$ satisfies the constrains. The parameters are optimized using Adam [13] with a fixed learning rate 0.001.

Table 1 Statistics of the real-world networks

	Cora	Citeseer	Pubmed-Diabetes
$\|V\|$	2708	3312	19,717
$\|E\|$	5429	4732	44,338
Avg. degree	4.01	2.86	4.50
# labels	6	6	3

5 Experimental results

5.1 Experiment Setup

Datasets Experiments are held on 3 real-world datasets[1] to evaluate the performances of the proposed model. These datasets are Citeseer, Cora and Pubmed-Diabetes. The detail information of these datasets is given in Table 1: nodes of Citeseer and Cora have 6 labels, and nodes of Pubmed-Diabetes have 3 labels. Compared with Cora (average degree 4.01) and Pubmed-Diabetes (average degree 4.50), Citeseer (average degree 2.86) is more sparse. These three datasets are pre-processed to be connected networks. Since the scale of Pubmed-Diabetes (19,717 nodes) is much larger than the other two datasets, we sample it into a small network with around 2, 000 nodes and the average degree is around 4.5.

Node similarity measurements Eleven node similarity measurements are used in this work to represent the network structural information. The detailed information of the node similarities is given in Table 2. The common neighbors method(CN) [25] is a typical local similarity-based approach which measures two nodes' similarity by the number of their one-hop common neighbors. The Adamic-Adar index(AA) [25], the resource allocation index(PA) [15] and resource allocation based on common neighbor interactions (RA-CNI) [15] improve CN by differentiating the influences the node's on-hop neighbors. The Jaccard index method [25] (JC) improves CN by measuring the ratio of two nodes' one-hop common neighbors over their complete one-hop neighbor set. The Salton index method(SA) [15], also known as the cosine similarity, and the Sorensen index method(SO) [15] improve JC by using different measurements of each node's one-hop complete neighbor set. In practical, the node similarity measured by SA is approximately twice the value of JC. SO is less sensitive to outliers compared with JC. The hub promoted index method(HPI) [15] measures the node similarity by the ratio of two nodes' one-hop common neighbors over the minimum of one node's one-hop neighbors, while the hub depressed index method(HDI) [15] measures the node similarity by the ratio of two nodes' one-hop common neighbors over the maximum of one node's one-hop neighbors. HPI discourages link formation between hub nodes and encourage link formation between low-degree nodes and hub nodes. This is opposite to HDI. The local Leicht-Holme-Newman index(LLHN) [15] is defined as the ratio of actual paths of length two between two nodes and a value proportional to the expected number of paths of length two between them. The preferential attachment index(PA) [15] is based on the phenomena that many real network node degrees follow a power law distribution. So if the degree of two nodes is large, they are more likely to form a link.

[1] https://linqs.soe.ucsc.edu/data

Table 2 Node similarity measurements used in this work

#	Name	Definition
1	JC	$S_{JC}(v_i, v_j) = \frac{\lvert\Gamma(v_i)\cap\Gamma(v_j)\rvert}{\lvert\Gamma(v_i)\cup\Gamma(v_j)\rvert}$
2	PA	$S_{PA}(v_i, v_j) = \lvert\Gamma(v_i)\rvert\lvert\Gamma(v_j)\rvert$
3	SA	$S_{SA}(v_i, v_j) = \frac{\lvert\Gamma(v_i)\cap\Gamma(v_j)\rvert}{\sqrt{\lvert\Gamma(v_i)\rvert\lvert\Gamma(v_j)\rvert}}$
4	CN	$S_{CN}(v_i, v_j) = \lvert\Gamma(v_i)\cap\Gamma(v_j)\rvert$
5	SO	$S_{SO}(v_i, v_j) = \frac{2\lvert\Gamma(v_i)\cap\Gamma(v_j)\rvert}{\lvert\Gamma(v_i)\rvert+\lvert\Gamma(v_j)\rvert}$
6	LLHN	$S_{LLHN}(v_i, v_j) = \frac{\lvert\Gamma(v_i)\cap\Gamma(v_j)\rvert}{\lvert\Gamma(v_i)\rvert\lvert\Gamma(v_j)\rvert}$
7	HPI	$S_{HPI}(v_i, v_j) = \frac{\lvert\Gamma(v_i)\cap\Gamma(v_j)\rvert}{min\left(\lvert\Gamma(v_i)\rvert,\lvert\Gamma(v_j)\rvert\right)}$
8	HDI	$S_{HDI}(v_i, v_j) = \frac{\lvert\Gamma(v_i)\cap\Gamma(v_j)\rvert}{max\left(\lvert\Gamma(v_i)\rvert,\lvert\Gamma(v_j)\rvert\right)}$
9	RA	$S_{RA}(v_i, v_j) = \sum_{v\in\Gamma(v_i)\cap\Gamma(v_j)} \frac{1}{\lvert\Gamma(v)\rvert}$
10	AA	$S_{AA}(v_i, v_j) = \sum_{v\in\Gamma(v_i)\cap\Gamma(v_j)} \frac{1}{log\,\lvert\Gamma(v)\rvert}$
11	RA-CNI	$S_{RACNI}(v_i, v_j) = \sum_{v\in\Gamma(v_i)\cap\Gamma(v_j)} \frac{1}{\lvert\Gamma(v)\rvert} + \sum_{(v_k,v_l)\in E,\lvert\Gamma(v_k)\rvert<\lvert\Gamma(v_l)\rvert,v_k\in\Gamma(v_i),v_l\in\Gamma(v_j)} \left(\frac{1}{\lvert\Gamma(v_k)\rvert}-\frac{1}{\lvert\Gamma(v_k)\rvert}\right)$

*v denotes node

*E is the edge set

*min (·) is the minimal function

*log (·) is the log function

*max (·) is the maximal function

*Γ(·) is the neighborhood set of node

Baseline methods Four of the most popular network embedding models are used as the baselines of this work:

- **DeepWalk**: DeepWalk [17, 18] uses random walk to get the node sequence and then Skip-Gram algorithm is used to learn the node representation vectors.
- **Node2Vec**: Node2Vec [8, 18] uses improved random walk step to get the node sequence and then uses the skip-Gram algorithm to learn the node representation vectors.
- **LINE**: LINE [18, 23] is a popular network embedding methods which considers the first and second order proximities information.
- **GraRep**: GraRep [4, 18] carries out matrix factorization based on different sized random walk.

Parameter settings The dimension of the latent representation is set to be 16 for all datasets, i.e., $d = 16$. The parameters of the baseline methods use default settings.[2] For the proposed model, the dimension of each single view is 16, the dimension of common representation is $\frac{16}{2}$,

[2] https://github.com/thunlp/openne

and the dimension of the Gaussian distribution in view-specific representation is $\frac{16}{4}$. In view-specific representation, we use a two hidden layer $\{3000, 600\}$ neural network.

5.2 Performance Evaluation

The network embedding performances are verified on node classification task, in which Support Vector Machine [14] is used to classify nodes. Macro-F1 score and Micro-F1 score are used to evaluate the node classification performances. For each dataset, different ratio of the labelled nodes are randomly selected as the training data, and the rest of nodes are used as the test data. The results are averaged over 50 different runs by sampling different training data.

Let MvNR represent the proposed multi-view network embedding model, node classification performances with MvNR is given in Figure 1. It is shown that: (1) Node classification performances with MvNR are superior to node classification performances with baseline methods. Both Micro-F1 score and Macro-F1 score by using MvNR are higher than using baseline methods on all three experimental datasets. (2) If the ratio of training set is low, node classification performances with MvNR are significantly higher than node classification performances with baseline methods. For example, when the training ratio is 0.1, compared with the best Micro-F1 score and Macro-F1 score achieved by baseline methods, Micro-F1 score and Macro-F1 score by using MvNR are about 15% and 25% higher on Citeseer dataset, about 30% and 40% higher on Cora dataset, and about 30% and 40% higher on Pubmed-Diabetes dataset. (3) Node classification performances are less sensitive to the ratio of training set by using MvNR. Baseline methods, especially random walk based methods Deepwalk and

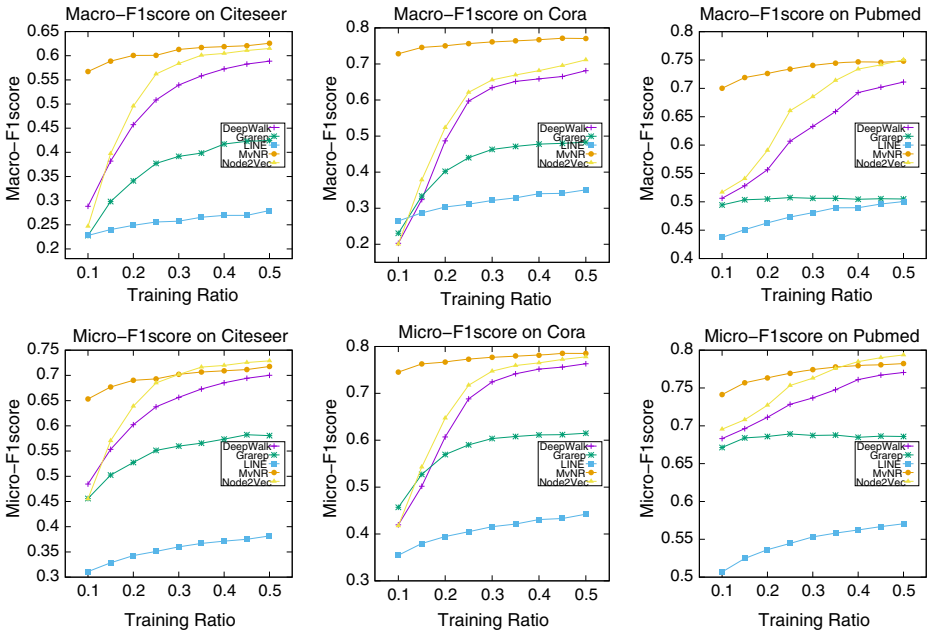

Figure 1 Node classification performances with the proposed multi-view network embedding model

Node2Vec, are sensitive to the ratio of training set: node classification performances decrease significantly with the decrease of training sample ratio. Using MvNR, there is only slightly change on node classification performances with the decrease of training sample ratio.

Deepwalk and Node2Vec preserve global node similarities between the active node and the nodes selected with co-occurrence probability when randomly walked in the network. LINE and GraRep preserve local node similarities, in which LINE measures the node similarity between the active node and the nodes with the first and the second order proximity, while GraRep measures the node similarity between the active node and the nodes with high order proximity. If the ratio of the training sample is low, it is hard describe the overall network structure by node similarities used in the baseline methods. This leads to their limited node classification performances. Since MvNR preserves network information with node similarity ensemble, the involved nodes similarities reflect the network structural information from different aspects of view. It can describe the overall network structure with limited number of training samples. This contributes to its superior node classification performances with low ratio of training set.

Figure 2 illustrates the comparison of node classification performances with MvNR and the average node classification performances preserving each node similarity listed in Table 2. It is shown that node classification performances with MvNR are significantly better than node classification performances preserving single node similarity. Micro-F1 score and Macro-F1 score by using MvNR are about 6% and 10% higher on Citeseer dataset, about 8% and 12% higher on Cora dataset, and about 4% and 8% higher on Pubmed-Diabetes dataset. Table 3 gives the detail information of node classification performances preserving each single node similarity respectively on Pubmed-Diabetes dataset. By preserving single node similarity, the highest Micro-F1 score and Macro-F1 score are 0.765 and 0.716 respectively, while the lowest Micro-F1 score and Macro-F1 score are 0.607 and 0.469 respectively. The selected node

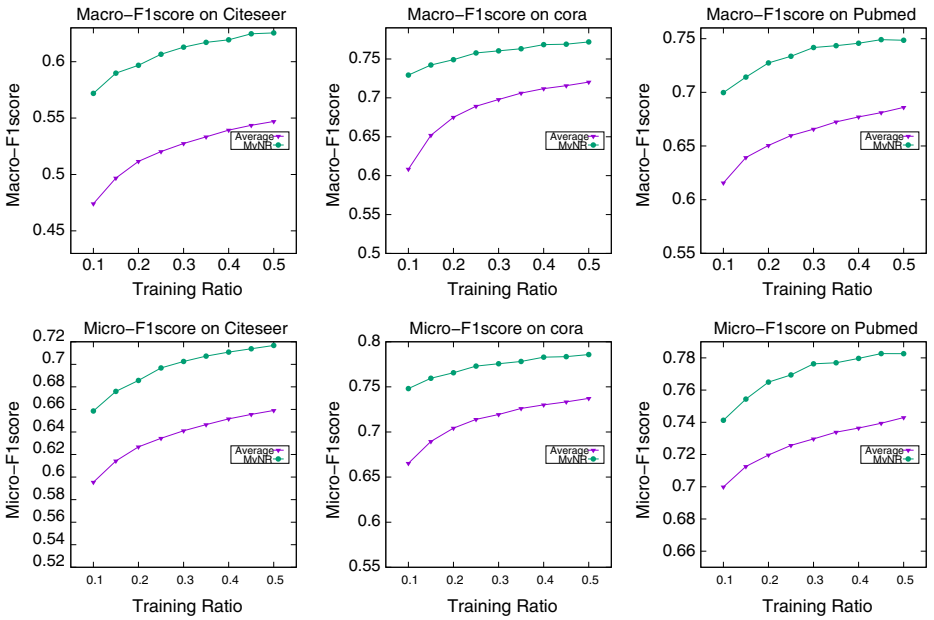

Figure 2 Comparison of node classification performances with the proposed model and the average node classification performances preserving all possible single node similarity

Table 3 Node classification performances preserving single node similarity and node similarity ensemble on Pubmed-Diabetes dataset

Metric	Node similarity	Training Ratio				
		10%	20%	30%	40%	50%
Micro-F1	AA	0.727	0.740	0.751	0.758	0.765
	CN	0.721	0.741	0.749	0.754	0.762
	HDI	0.721	0.737	0.743	0.747	0.750
	HPI	0.717	0.733	0.739	0.753	0.755
	JC	0.708	0.725	0.734	0.741	0.742
	LLHN	0.682	0.713	0.728	,0.738	0.745
	PA	0.607	0.630	0.640	0.648	0.652
	RA	0.733	0.744	0.753	0.759	0.765
	RA-CNI	0.693	0.716	0.728	0.740	0.744
	SA	0.702	0.725	0.736	0.748	0.751
	SO	0.690	0.706	0.718	0.726	0.731
	Average	0.700	0.719	0.729	0.737	0.742
	MvNR	0.741	0.765	0.776	0.780	0.783
Macro-F1	AA	0.651	0.676	0.693	0.706	0.716
	CN	0.645	0.680	0.692	0.700	0.710
	HDI	0.649	0.678	0.687	0.693	0.697
	HPI	0.649	0.671	0.682	0.700	0.705
	JC	0.636	0.664	0.679	0.689	0.692
	LLHN	0.619	0.668	0.686	0.696	,0.705
	PA	0.469	0.510	0.530	0.543	0.549
	RA	0.655	0.674	0.688	0.699	0.711
	RA-CNI	0.597	0.641	0.660	0.675	0.683
	SA	0.609	0.652	0.672	0.688	0.692
	SO	0.606	0.635	0.652	0.666	0.674
	Average	0.617	0.650	0.666	0.678	0.685
	MvNR	0.700	0.727	0.742	0.746	0.749

similarities contribute to reasonable node classification performances compared with baseline methods shown in Figure 1. By preserving node similarity ensemble with common latent space and view-specific latent spaces, MvNR contributes to better node classification performances compared with preserving any single node similarity, and better node classification performances compared the average performances of preserving all possible node similarities.

To sum up, by preserving node similarity ensemble, the proposed multi-view network embedding model contributes to better node classification performances compared with network embedding models preserving single node similarity or simple combination of node similarities. It is only slightly influenced by the ratio of training set, and it contributes to valuable node classification performances with very low ratio of training set.

6 Conclusions

This work proposes a novel multi-view network embedding model with node similarity ensemble. Instead of preserving single node similarity in the network embedding, the proposed method generates multiple views of latent spaces, preserving various node similarity sets. To maximize the involved network information while minimize the information redundancy, common representation is extracted from multiple views to merge with multiple view-specific representations. The common representation of multiple views is extracted with a

CCA based model, and view-specific representations are extracted with a neural network based model analyzing Gaussian distributions of nodes. Node classification performances with the proposed method are superior to node classification performances with existing network embedding models, especially when there is only very limited number of training samples.

Acknowledgements This research was supported by National Natural Science Foundation of China (Grant No. 61672284), Natural Science Foundation of Jiangsu Province (Grant No. BK20171418), China Postdoctoral Science Foundation (Grant No. 2016 M591841), Jiangsu Planned Projects for Postdoctoral Research Funds (No. 1601225C). The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through research group no. RGP-VPP-264.

References

1. Bai, X., Cao, H., Zhao, T.: Improving vector space word representations via kernel canonical correlation analysis. ACM Trans. Asian & Low-Resource Lang. Inf. Process. **17**(4), 29:1–29:16 (2018)
2. Bojchevski, A., Günnemann, S.: Deep gaussian embedding of attributed graphs: unsupervised inductive learning via ranking. CoRR **abs/1707.03815** (2017)
3. Bu, Y., Zou, S., Liang, Y., Veeravalli, V.V.: Estimation of KL divergence: optimal minimax rate. IEEE Trans. Inf. Theory. **64**(4), 2648–2674 (2018)
4. Cao, S., Lu, W., Xu, Q.: Grarep: learning graph representations with global structural information. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, pp. 891–900. ACM, Melbourne (2015)
5. Dong, Y., Chawla, N.V., Swami, A.: Metapath2vec: scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD 2017, pp. 135–144. ACM, Halifax (2017)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, D.M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, vol. 9, pp. 249–256. JMLR.org, Sardinia (2010)
7. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowl.-Based Syst. **151**, 78–94 (2018)
8. Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD 2016, pp. 855–864. ACM, San Francisco (2016)
9. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. IEEE Data Eng. Bull. **40**(3), 52–74 (2017)
10. He, Y., Liu, J.N., Hu, Y., Wang, X.: OWA operator based link prediction ensemble for social network. Expert Syst. Appl. **42**(1), 21–50 (2015)
11. He, Y., Wang, C., Jiang, C.: Discovering canonical correlations between topical and topological information in document networks. IEEE Trans. Knowl. Data Eng. **30**(3), 460–473 (2018)
12. Huang, X., Li, J., Hu, X.: Label informed attributed network embedding. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, pp. 731–739. ACM, Cambridge (2017)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR **abs/1412.6980** (2014)
14. Li, C., Wang, S., Yang, D., Li, Z., Yang, Y., Zhang, X., Zhou, J.: PPNE: property preserving network embedding. In: Database Systems for Advanced Applications - 22nd International Conference, *DASFAA 2017*, vol. 10177, pp. 163–179. Springer, Suzhou (2017)
15. Martínez, V., Berzal, F., Talavera, J.C.C.: A survey of link prediction in complex networks. ACM Comput. Surv. **49**(4), 69:1–69:33 (2017)
16. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (eds.) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD 2016, pp. 1105–1114. ACM, San Francisco (2016)
17. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, pp. 701–710. ACM, New York (2014)

18. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, pp. 459–467. ACM, Marina Del Rey (2018)
19. Qu, M., Tang, J., Han, J.: Curriculum learning for heterogeneous star network embedding via deep reinforcement learning. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM '18, pp. 468–476. ACM, New York (2018)
20. Rupnik, J., Muhic, A., Leban, G., Fortuna, B., Grobelnik, M.: News across languages - cross-lingual document similarity and event tracking (extended abstract). In: C. Sierra (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 5050–5054. ijcai.org, Melbourne (2017)
21. Sheikh, N., Kefato, Z., Montresor, A.: gat2vec: representation learning for attributed graphs. Computing. 1–23 (2018)
22. Shi, Y., Gui, H., Zhu, Q., Kaplan, L.M., Han, J.: Aspem: Embedding learning by aspects in heterogeneous information networks. In: Proceedings of the 2018 SIAM International Conference on Data Mining, SDM 2018, pp. 144–152. SIAM, San Diego (2018)
23. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, pp. 1067–1077. ACM, Florence (2015)
24. Yuan, W., He, K., Guan, D., Han, G.: Edge-dual graph preserving sign prediction for signed social networks. IEEE Access. **5**, 19383–19392 (2017)
25. Yuan, W., He, K., Guan, D., Zhou, L., Li, C.: Graph kernel based link prediction for signed social networks. Information Fusion. **46**, 1–10 (2019)
26. Yuan, W., He, K., Han, G., Guan, D., Khattak, A.M.: User behavior prediction via heterogeneous information preserving network embedding. Futur. Gener. Comput. Syst. **92**, 52–58 (2019)
27. Zhang, Y., Zhang, J., Pan, Z., Zhang, D.: Multi-view dimensionality reduction via canonical random correlation analysis. Frontiers Comput. Sci. **10**(5), 856–869 (2016)

Affiliations

Weiwei Yuan[1,2] · Kangya He[1] · Chenyang Shi[1] · Donghai Guan[1] · Yuan Tian[3] · Abdullah Al-Dhelaan[4] · Mohammed Al-Dhelaan[4]

Weiwei Yuan
yuanweiwei@nuaa.edu.cn

Kangya He
kangyahe@gmail.com

Chenyang Shi
chenyangshi1996@gmail.com

Yuan Tian
ytian@njit.edu.cn

Abdullah Al-Dhelaan
dhelaan@ksu.edu.sa

Mohammed Al-Dhelaan
mdhelaan@ksu.edu.sa

[1] College of Computer Science & Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

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

[3] School of Computer Engineering, Nanjing Institute of Technology, Nanjing, China

[4] Department of Computer Science, King Saud University, Riyadh, Saudi Arabia