From Properties to Links: Deep Network Embedding on Incomplete Graphs

Dejian Yang
Beihang University
dejianyang@buaa.edu.cn

Senzhang Wang
Nanjing University of Aeronautics and Astronautics
szwang@nuaa.edu.cn

Chaozhuo Li*
Beihang University
lichaozhuo@buaa.edu.cn

Xiaoming Zhang
Beihang University
yolixs@buaa.edu.cn

Zhoujun Li
Beihang University
lizj@buaa.edu.cn

ABSTRACT

As an effective way of learning node representations in networks, network embedding has attracted increasing research interests recently. Most existing approaches use shallow models and only work on static networks by extracting local or global topology information of each node as the algorithm input. It is challenging for such approaches to learn a desirable node representation on incomplete graphs with a large number of missing links or on dynamic graphs with new nodes joining in. It is even challenging for them to deeply fuse other types of data such as node properties into the learning process to help better represent the nodes with insufficient links. In this paper, we for the first time study the problem of network embedding on incomplete networks. We propose a Multi-View Correlation-learning based Deep Network Embedding method named MVC-DNE to incorporate both the network structure and the node properties for more effectively and efficiently perform network embedding on incomplete networks. Specifically, we consider the topology structure of the network and the node properties as two correlated views. The insight is that the learned representation vector of a node should reflect its characteristics in both views. Under a multi-view correlation learning based deep autoencoder framework, the structure view and property view embeddings are integrated and mutually reinforced through both self-view and cross-view learning. As MVC-DNE can learn a representation mapping function, it can directly generate the representation vectors for the new nodes without retraining the model. Thus it is especially more efficient than previous methods. Empirically, we evaluate MVC-DNE over three real network datasets on two data mining applications, and the results demonstrate that MVC-DNE significantly outperforms state-of-the-art methods.

*Corresponding author: Chaozhuo Li(lichaozhuo@buaa.edu.cn)

CIKM'17, , November 6–10, 2017, Singapore.

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132975

CCS CONCEPTS

• **Information systems → Data mining**; • **Computing methodologies → Dimensionality reduction and manifold learning**;

KEYWORDS

Network Embedding, Incomplete Graph, Deep Learning

1 INTRODUCTION

Networks are ubiquitous nowadays, such as social networks (Twitter, Flickr), paper citation networks (DBLP, Google Scholar), knowledge graphs (Freebase, Wikipedia) and communication networks (Email). Mining valuable knowledge from networks can facilitate many real applications in practice. For example, clustering the users in social networks into different communities can help advertisers better perform online advertisement targeting [3]. A fundamental problem in network mining is how to learn a desirable representation vector for each node [5]. As an effective way to embed each node into a low-dimensional continuous feature vector, network embedding has been extensively studied recently. The learned high-quality node representation vectors are fundamental to perform many data mining and machine learning tasks, such as classification [22], social recommendation [8] and link prediction [16].

Most existing network embedding models such as DeepWalk [19], LINE [23], GraRep [5] and Node2Vec [10] are shallow models. They adopt a well-designed objective function to capture the topology information of the input network, while the node representation vectors are considered as parameters that need to be learned in the objective function. Such network structure based shallow models purely utilize the network topology information and encode the local and global structures of the nodes into their representation vectors. To achieve better embedding performance on the networks with sparse links, some recent works [13, 14, 28, 29] try to incorporate node properties and community structures as auxiliary information into the learning of the network embedding.

Although remarkable efforts have been made on node representation learning on static complete networks, how to effectively and efficiently perform network embedding on incomplete networks is still not well studied. Typically, parts of the links among the nodes in an incomplete graph can be missing, leading to the challenge for existing approaches which purely or largely rely on the topology information of the networks to learn a promising network embedding result. Meanwhile, a more challenging case is when

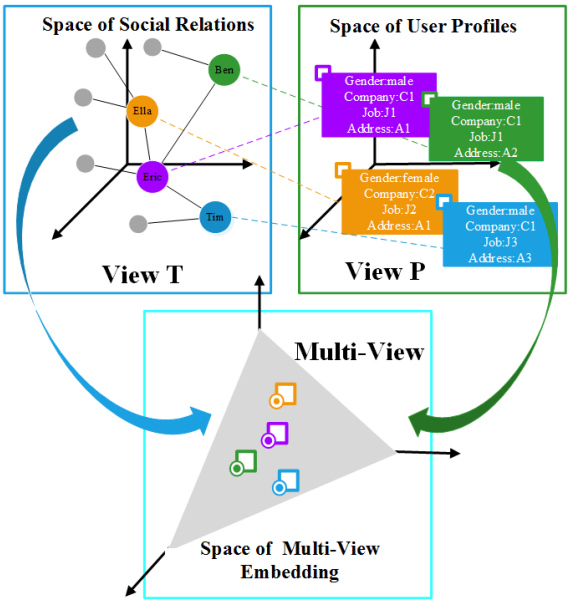


Figure 1: An illustration of multi-view data in the initial spaces in social networks and their corresponding representations in the multi-view embedding space.

the graph is dynamic with new nodes joining in. Previous models face with the challenges of efficiency and effectiveness in learning representations for such new nodes with very limited or even no links. To learn the representation vectors for new nodes, tradition models need to retrain the model which is very time consuming. For example, when dealing with the new nodes, the popular DeepWalk algorithm needs to regenerate the entire nodes sequences as the new input, and then retrains the Skip-Gram model [19].

In this paper, we for the first time study the problem of network embedding on incomplete networks. Given a node in a network, besides the links between the node to others, it can be also associated with a set of node properties such as the user profiles of a user in social networks (as shown in the upper part of Figure 1) or the metadata of a paper in citation networks. Such node properties are highly correlated to links in terms of node similarity, and thus they are important complementary information to the structural topology in network embedding. For example, as shown in Figure 1, two connected users are more likely to have similar user profiles than two users that are far away from each other topologically due to the characteristic of homophily [1, 7, 17]. Motivated by this idea, we propose to introduce node properties into network embedding on incomplete graphs. The node properties potentially encode different types of but highly correlated information to the network topology, and integrate them into a unified learning framework is expected to achieve a better performance. Although previous work [29] tried to utilize texts associated with each node to improve the performance of network embedding, that method used the two types of data in a shallow way. It is difficult for [29] to learn the highly nonlinear correlations between the texts and links by considering the two types of data separately and combining them shallowly.

To address above mentioned problem, in this paper we propose a deep network embedding framework on incomplete graphs with only partial links available, aiming to more efficiently and effectively learn representation vectors by considering both the network topology and node properties. We view the topology structure and the node properties as two correlated modalities of the network, and propose a Multi-View Correlation learning based Deep Network Embedding method, named MVC-DNE. The insight is that the learned representation vector of a node should reflect its characteristics in both views (as shown in the lower part of Figure 1). Specifically, we first utilize the deep autoencoder to obtain the latent representations in each single view. Note that the representations are calculated by deep autoencoders, and thus it can be considered as a map function. Then we design two frameworks to learn the correlations in multiple views in the encoding stage and decoding stage, respectively. Besides self-view learning implemented by the deep autoencoder in each single view, our method considers the features of one view as labels to guide the encoding process of another view. The self-view and cross-view learnings are able to capture the high level associations between the two views.

We summarize the main contributions of this paper as follows:

- We for the first time study the novel problem of network embedding on incomplete graphs. We introduce node properties and propose a deep network embedding framework MVC-DNE to address the challenge of sparsity in the network structure view.
- We design two frameworks to learn the correlations of node representations in the two views. The consistency captured by cross-view learning from one view is able to complete or refine the features of another view. By learning a representation vector mapping function, our models are much more efficient to generate representations for new nodes without needing to retrain the model.
- Extensively, we evaluate MVC-DNE on three real network datasets through multi-class classification and link prediction. The results show the superior performance of MVC-DNE by comparison with state-of-the-art baseline methods.

The rest of this paper is organized as follows. Section 2 summarizes the related works. Then we formally define the studied problem in Section 3. Section 4 introduces the proposed multi-view deep network embedding framework in details. Section 5 presents the experimental results. Finally we conclude this work in section 6.

2 RELATED WORK

2.1 Network Embedding

Network Embedding aims to project each node in a network into a distributed representation. Most earlier works on this problem [2, 4, 11, 15, 20, 25] represent the network as an affinity matrix and then extract the leading eigenvectors as the representations of nodes by using matrix factorization techniques. For example, IsoMAP [25] uses the feature vectors of the nodes to construct an affinity graph, and then represent the nodes of the network with the solved leading eigenvectors.

Recently, DeepWalk [19], a online learning method, utilizes random walks to transform the structural information into a set of node sequences. Then skip-gram is adopted to learn the network

representations like word embedding. It has been proven to be effective for preserving the second-order proximity of the nodes in the network. LINE [23] is the first to use first-order proximity and second-order proximity, which preserves both the local and global structure information of network. Correspondingly, [23] takes the direct linking relations which represent the first order proximity into account and proposes a faster algorithm to perform network embedding. To overcome the limitation of prior works in failing to offer flexibility in sampling nodes from a network [19, 23], Node2Vec [10] designs a much more flexible objective that is not tied to a particular sampling strategy and provides parameters to tune the explored search space. In addition, [26] adopt max-margin principle based on DeepWalk and incorporate labeling information into vertex representations.

However, the above discussed methods can all be regarded as shallow models, and they are not effective to capture the highly nonlinear structural information of the networks. To address this problem, SDNE [27] is the first deep model which utilizes the modified deep atuoencoder. SDNE can preserve both the first and second order proximity in the network with nonlinear structure relations captured by the sigmoid functions and the multiple layers. However, SDNE can not effectively handle multi-view data such as node properties. TADW [29] for the first time incorporates the citation relations and the rich text information of paper to learn a text-associated representations for each node in the citation network. As TADW is based on the matrix factorization technique, it is very time consuming to process large networks. In addition, it is difficult for TADW to achieve a satisfactory performance when the network structure is sparse or when the text information is insufficient due the interdependence between topology structure and the texts.

3 PROBLEM DEFINITION

In this section, we will give a formal definition of the studied problem. We first give some notations as follows. Network G with each node associated with a set of properties is defined as $G = (V, T, P)$, where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ denotes the nodes in the network. $T \in \mathbb{R}^{|V|\times|V|}$ represents the adjacency matrix and $P \in \mathbb{R}^{|V|\times|P|}$ is the property matrix of nodes. Here $|V|$ and $|P|$ represent the dimensions of the adjacency matrix and the node properties, respectively. Note that we define the dimension of the node properties as $\sum_{i=1}^{m} n_i$, where m is the size of the property categories and n_i is the number of possible discrete values of the ith property. With such a definition, the element value of the node property vector is in $\{0, 1\}$ such that it can be processed by our proposed deep model. With the above defined notations, we formally define the studied problem as follows.

Definition 3.1. ***Network Embedding on Incomplete Graphs with Multi-View Data.*** *Given a network $G = (V, T, P)$, with T denoting the network structure view data, and P denoting the node property view data as shown in Figure 1, network embedding aims to learn a function f, which maps the node of both view features into a continuous low-dimensional vector in the following two cases: 1) the network G is incomplete with only partial links in the T view available; 2) the network G is dynamic with new nodes joining in, and the links of the new nodes are very sparse. That is for each node $v_i \in V$ in both cases, our task is to learn such a distributed representation $y_i \in \mathbb{R}^{2d} = f(xt_i, xp_i)$ where $d \ll |V|$ is the output dimension of each view, xt_i and xp_i represent the input vector of two views respectively.*

In our problem, T and P represent the information of a network from different views. Most existing methods adopt a well-designed objective function to perform network embedding only with the topology structure preserved. In an incomplete network, the links or properties of nodes may be missing. It is challenging for previous methods to learn a satisfactory representations in such a case. To address this problem, one solution is to integrate the two views and deeply mine their latent correlations in the shared embedding space. To this aim, next we will introduce how to integrate the topology structure view and the node property view under a multi-view deep correlation learning framework to better performa network embedding on incomplete networks.

4 DEEP NETWORK EMBEDDING VIA MULTI-VIEW CORRELATION LEARNING

In this section, we will first introduce how to adopt deep autoencoders to perform network embedding in the network structure view. Based on it, we will next propose two learning frameworks, called MVC-DNE$_{DB}$ and MVC-DNE$_{EB}$ to perform network embedding on incomplete graphs by incorporating node properties and learning the correlations from the two views.

4.1 AutoEncoder-based Deep Network Embedding Model in a Single View

The traditional deep autoencoder has been proven to be effective in dimension reduction [21]. SDNE is the first work that utilizes autoencoder based deep network embedding model and achieves promising performance on static and complete graphs [27]. Thus here we propose to utilize the deep autoencoder to perform network embedding in a single view of network. The model, named SV-DNE, maps the input data of one view to distributed representations through the encoder part, and then use the decoder to reconstruct the input data. Formally, given the input data x, the representations in each hidden layer are defined as follows:

$$y^{(1)} = \sigma(W^{(1)} \cdot x + b^{(1)}) \tag{1}$$

$$y^{(k)} = \sigma(W^{(k)} \cdot y^{(k-1)} + b^{(k)}), k = 2, \ldots, K \tag{2}$$

where K denotes the number of layers, $W^{(k)}$ and $b^{(k)}$ denote the k-th layer weight matrix and biases respectively. We adopt the sigmoid activation function to capture the complex relations of features, i.e. the highly nonlinear structure in view T. When $y^{(K)}$ is obtained, we can reverse the above process to calculate the reconstructed input data $\hat{x}$. By minimizing the reconstruction error between x and $\hat{x}$, the node can be represented by a d-dimensional vector $y^{(K)}$ with the information in the corresponding view preserved. The loss function of network embedding in a single view is as follows.

$$L(x; \theta) = \sum_{i=1}^{|V|} \|\hat{x_i} - x_i\|_2^2 \tag{3}$$

where $\theta = \{W^{(k)}, b^{(k)}, W^{\hat{(k)}}, b^{\hat{(k)}}\}_{k=1}^{K}$ denotes all the parameters.

However, a major problem of this method is that all features are treated equally during the reconstruction process. Study in [27] has shown that paying more attention to the non-zero elements of the feature vector can learn a better representation as the input feature vector x has far more zero elements. By taking this idea into consideration, the loss function is redefined as follows.

$$L(x;\theta) = \sum_{i=1}^{|V|} \|(\hat{x_i} - x_i) \odot w_i\|_2^2 \tag{4}$$

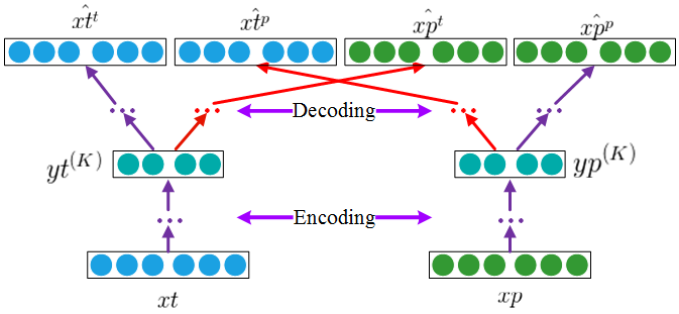
where w_i is a weight vector with the same dimension as x, and each element $w_{i,j} = \beta > 1$ if $x_{i,j} > 0$, otherwise $w_{i,j} = 1$. Note that β varies for different views. Therefore, this model is able to represent the nodes with data information denoted by the non-zero elements fully preserved, i.e. the topology structure or the nodes properties.

As a typical unsupervised learning method, the above SV-DNE utilizes the non-zero elements to supervise the learning process. We call this feature encoding self-view learning. The hidden vectors in the Kth layer are the output node representation in a single view.

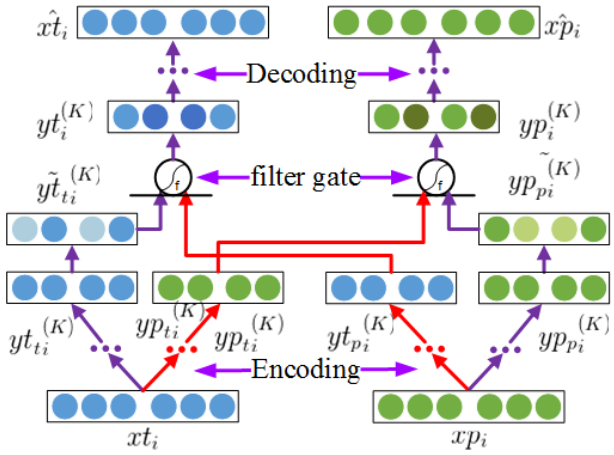
4.2 AutoEncoder-based Deep Network Embedding in Multiple Views

Based on the above proposed network embedding with deep autoencoder framework in a single view, we next will introduce how to construct a deep network embedding model with multi-view data by incorporating node properties and network structure. A straightforward method is to build two subnetworks SV-DNE$_T$ on view T and SV-DNE$_P$ on view P respectively, and then concatenate $yt^{(K)}$(the output representation in view T) and $y^{(K)}$(the output representation in view P) as the final $2d$-dimensional representations. However, this direct feature concatenation of SV-DNE$_T$ and SV-DNE$_P$ is not capable of capturing the correlations between two views. This combination does not work well when one view data is missing. Besides, the correlations is of the essence to handle the sparse and incomplete networks, such as new nodes embedding. To address this problem, we propose the following two models.

MVC-DNE$_{DB}$. As the features in the network structure view T and the node property view P can be highly correlated, the input features of one view can encode some shared latent information reflected by the input of the other view. Inspired by this idea, we propose a framework named MVC-DNE$_{DB}$ which learns the cross-view correlations by the decoding operation. This framework aims to mine the latent information of the other view and make the learned representations in each view unified in both views. As shown in Figure 2(a), the Kth-layer representation $yt^{(K)}$ ($yp^{(K)}$) not only reconstructs the input of view T (view P) denoted as self-view decoding, but also reconstructs the input of another view denoted as cross-view decoding. Thus the hidden vectors($yt^{(K)}$ and $yp^{(K)}$) is able to represent the information of both views. The red arrows in Figure 2(a) denote the cross-view learning. Specifically, MVC-DNE$_{DB}$ utilizes xt (xp) to guide and refine the encoding process in view P (T). Therefore, we sum the reconstruction errors in both views as the final objective function:



(a) Decoding-based MVC-DNE



(b) Encoding-based MVC-DNE

Figure 2: Two implementations of MVC-DNE: MVC-DNE$_{DB}$ and MVC-DNE$_{EB}$. The purple arrows indicate self-view learning, and the red arrows indicate the cross-view learning, respectively.

$$L_{DB}(xt, xp; \theta) = L_t(xt, xp; \theta) + L_p(xt, xp; \theta) \tag{5}$$

$$L_t(xt, xp; \theta) = \sum_{i=1}^{|V|}((1-\alpha)\|xt_i - \hat{xt_i^t}\|_2^2 + \alpha\|xt_i - \hat{xt_i^p}\|_2^2) \tag{6}$$

$$L_p(xt, xp; \theta) = \sum_{i=1}^{|V|}((1-\alpha)\|xp_i - \hat{xp_i^p}\|_2^2 + \alpha\|xp_i - \hat{xp_i^t}\|_2^2) \tag{7}$$

Here $\hat{xt_i^t}$ and $\hat{xt_i^p}$ denote the reconstruction outputs of xt_i decoded by the hidden representations $yt_i^{(K)}$ and $yp_i^{(K)}$ respectively, and one can also get the meanings of $\hat{xp_i^t}$ and $\hat{xp_i^p}$ similarly. α is a parameter to balance the importance of the self-view and cross-view reconstruct errors.

By minimizing the above four reconstruction errors, the inputs of both views are projected into a consistent latent semantic space.

Thus the representation in the hidden space of each view not only preserves the features of its own view but also encodes the information from the other view.

MVC-DNE$_{EB}$. Another possible solution is to refine or modify the node characteristics reflected in one view data by bring the information from the other view data. Following this idea, we propose a new model called MVC-DNE$_{EB}$. Different from MVC-DNE$_{DB}$ that learns the correlations of the two views in the decoding operation, MVC-DNE$_{EB}$ tries to learn the correlations in the encoding operation. Specifically, as illustrated in Figure 2(b), MVC-DNE$_{EB}$ adopts four types of encoders to represent the inputs of the two views. These four encoders are classified into the self-view encoding (purple arrow in Figure 2) and cross-view encoding (red arrow in Figure 2). The self-view encoding aims to preserve the features of its own view while the cross-view encoding aims to integrate the shared latent information of the other view. When obtaining the outputs $yt_t^{(K)}$, $yt_p^{(K)}$, $yp_p^{(K)}$, $yp_t^{(K)}$ of the four encoders, the final representations $yt^{(K)}$ in view T and $yp^{(K)}$ in view P can be calculated as follows.

$$yt^{(K)} = \tilde{yt_t^{(K)}} + z_t(yt_p^{(K)}, \tilde{yt_t^{(K)}}) \odot yt_p^{(K)} \tag{8}$$

$$yp^{(K)} = \tilde{yp_p^{(K)}} + z_p(yp_t^{(K)}, \tilde{yp_p^{(K)}}) \odot yp_t^{(K)} \tag{9}$$

where $\tilde{yt_t^{(K)}}$ and $\tilde{yp_p^{(K)}}$ denote the outputs of the drop-out operation of $yt_t^{(K)}$ and $yp_p^{(K)}$, which only works in the training step. To guide the cross-view encoding and avoid overfitting, we randomly drop out 50% of the latent representations. z_t (z_p) is a weight vector, which determines how much information of $yt_p^{(K)}$ ($yp_t^{(K)}$) flowing into the output representation $yt^{(K)}$ ($yp^{(K)}$).

$$z_j(x, C) = \sigma(W_j^{(X)} \cdot x + W_j^{(C)} \cdot C + b_j), j = t, p \tag{10}$$

Note that the combination of the two representations are calculated by a filter gate, which has been proven its significant performance in information fusion [9]. In the decoding operation, the latent representations $yt^{(K)}$ and $yp^{(K)}$ are used to reconstruct the initial input data xt and xp. We define the loss function of MVC-DNE$_{EB}$ as follows.

$$L_{EB}(xt, xp; \theta) = L_t(xt, xp; \theta) + L_p(xt, xp; \theta) \tag{11}$$

$$L_t(xt, xp; \theta) = \sum_{i=1}^{|V|}(\|xt_i - \hat{xt_i^t}\|_2^2 \tag{12}$$

$$L_p(xt, xp; \theta) = \sum_{i=1}^{|V|}(\|xp_i - \hat{xp_i^p}\|_2^2 \tag{13}$$

To minimize the reconstruction errors, the dropped representation of one view can only be learned from the other view, and thus the information of one view is enhanced and complemented by the information of the other view. This is especially helpful when one view information like the network structure is sparse while the other view information like node properties is sufficient. Finally, we concatenate $yt^{(K)}$ and $yp^{(K)}$ as the final multi-view representations in MVC-DNE$_{DB}$ and MVC-DNE$_{EB}$. Note that for a particular task, we can represent the hidden vectors $yt^{(K)}$ or $yp^{(K)}$ as the learned features. For example, in link prediction, $yt^{(K)}$ may achieve better performance than the concatenated vectors because structural information may play a more important role.

4.3 Analysis and Discussions

Some analysis and discussions on MVC-DNE$_{DB}$ and MVC-DNE$_{EB}$ are presented as follows:

Embedding New Nodes. Many real-world networks are dynamic and it is practically important to quickly obtain reasonable representations for the new nodes with very limited or even no links. Compared to SNDE [27] that cannot effectively handle such new nodes, our models utilize the properties of nodes and learn the correlations between node properties and links, and thus obtain satisfactory representations for the new nodes.

A significant advantage of our modes is that our models adopt deep autoencoders to implement the map function f from the input to the representations in both views, denoted as the concatenation of $yt^{(K)}$ and $yp^{(K)}$ specifically. Thus, given the adjacency vector xt_{new} and property vector xp_{new} of a new node, the representation vector can be directly generated by the representation mapping function $y_{new} = f(xt_{new}, xp_{new})$ in time $O(1)$.

MVC-DNE$_{DB}$ vs MV-DNE$_{EB}$. Although MVC-DNE$_{DB}$ and MV-DNE$_{EB}$ are based on the similar idea, they conduct the cross-view correlation learning in different learning stages. MVC-DNE$_{DB}$ performs the cross-view learning in the decoding stage, and utilizes the input features of one view to guide the encoding of another view. The representation learning of each view relies on decoding the inputs of both views. MV-DNE$_{EB}$ performs the cross-view learning in the encoding stage, and implements the cross-view representation learning of one view through selecting the features of the other view that are highly correlated to the dropped out features of the view by the filter gate. The final output of MVC-DNE$_{DB}$ is a unified representation vector which fuses both view information; while the output of MV-DNE$_{EB}$ are two separate representation vectors with each one associated with one view.

5 EXPERIMENTS

In this section, we will evaluate the performance of our proposed models on three real network datasets through two tasks, multi-class classification and link prediction. We will first introduce the used datasets and the baseline methods, and then show and analyze the results under different experiment settings. Finally, we will conduct parameter sensitivity analysis.

We focus on evaluating the performance of the proposed models from the following three aspects:

(1) *whether the proposed models can effectively integrate multi-view data and outperforms previous shallow network embedding models on complete networks;*
(2) *whether the proposed models can learn a promising network embedding result on incomplete networks;*
(3) *whether the proposed models can efficiently and effectively generate representations for news nodes without retraining the model.*

Table 1: Statistics of the three datasets. (avg_t denotes the average number of links per node and avg_p denotes the average number of properties per node.)

Dataset	$\|V\|$	$\|E\|$	$\|P\|$	avg_t	avg_p	Categories
Citeseer	3,312	4,732	3,703	2.85	31.75	6
Google+	15,819	4,369,682	4,211	140.52	2.01	5
DBLP	35,750	143,848	6,295	8.05	6.38	8

5.1 Datasets

Our datasets include two academic paper citation networks Citeseer[1] [22] and DBLP[2] [24], and one social network dataset Google+[3] [12]. Table 1 shows the statistics of the three datasets after data preprocessing. We briefly introduce the datasets as follows.

- **Citeseer** is a paper citation network with around 3 thousand nodes. Each node represents a paper and each link indicates a citation relationship between two papers. The key words, abstract, and title are the property data of the paper. We use a binary adjacent vector of 3,312 dimensions to represent the structure features. The property features of each node are described by a binary vector of 3, 703 dimensions indicating the presence of the corresponding word. These papers are divided into 6 classes based on their topics or research fields.
- **Google+** is a social network, in which nodes refer to users and the edges denote the friend relationships among users. The user profiles, i.e. gender, institution, university and workplace, are the properties of the users. The job title is extracted as the job label of the user and we remove the nodes with no job title. Then we select the top 10 popular labels and cluster them into 5 classes as the final job category.
- **DBLP** contains 35,750 papers from 8 research domains. We choose the tile as the properties of each node. The nodes with no links are removed. Similar to Citeseer, we use a binary vector of 6,295 dimensions indicating the presence of the words in the property view to represent each node.

5.2 Baseline Methods

We compare our models with the following baselines. The first four baselines are state-of-the-art network embedding methods, and the other two are the variants of our proposed models.

- **DeepWalk** [19] is a shallow model. It combines random walks and skip-gram to learn network representations, which ensures the nodes with similar topology context are closer to each other in the projected latent space.
- **LINE** [23] is a also shallow network embedding model with the first order and second order proximity preserved. In this paper, we use LINE_{1st} and LINE_{2nd} as two baseline methods respectively.
- **TADW** [29] is the first shallow model incorporates rich text information of the network and adopts matrix factorization technique to obtain the node representations.

[1]http://linqs.cs.umd.edu/projects/projects/lbc/index.html
[2]https://cn.aminer.org/billboard/citation
[3]https://snap.stanford.edu/data/index.html

Table 2: Layer structures of the model on the three dataset

Dataset	layers in view $\|V\|$	layers in view $\|P\|$
Citeseer	3,312-100	3,703-100
Google+	15,819-300-100	4,211-100
DBLP	35,750-500-100	6,295-200-100

- **SDNE** [27] is the first deep network embedding method proposed recently. It uses the deep autoencoder to capture the no-linear relations and also designs an explicit objective function to preserve the local and global structure. SDNE can only handle the network structure information.
- **SV-DNE**$_P$ is a variation of our proposed method, which takes the property information of nodes as the input of SV-DNE to obtain the node representations. It only utilizes the property information of the nodes.
- **SDNE+SV-DNE**$_P$ is a naive combination of SDNE and SV-DNE$_P$. It concatenates the vectors learned by the two methods as the output of node representations.

5.3 Parameter Settings

In SDNE and our models, the deep autoencoders are used and the number of layers varies with the scale of networks. The details of neural network structure are shown in Table 2. The output dimension d is set as 100 in each view. The parameter β_T denoting the β in view T and β_T denoting the β in view P are tuned from 1 to 9, and we find 7 for β_T and 5 for β_P are reasonable settings after validation. We set the pre-training learning rate in SGD as 0.1 and fine-tuning learning rate in Adam as 0.001. The mini-batch size of optimization is fixed to 32.

In models LINE, DeepWalk and TADW, we follow the parameter settings in their papers, and set the negative samples as 5, the total training samples as 10 billion and the initial learning rate as 0.025, window size as 10, walk length as 40, walks per node as 40. The hyper-parameter parameter α in SDNE is fixed to 0.5.

5.4 Experiment Results on Complete Networks

In this section, we evaluate the performance of various models on complete graphs by conducting experiments on multi-class classification and link prediction tasks.

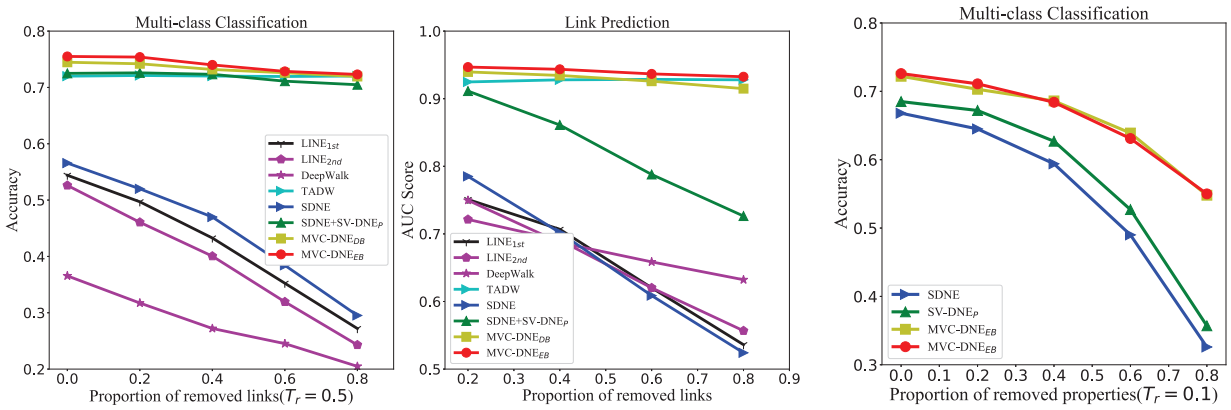
Performance evaluation on multi-class classification. We use the learned node representations as the input features of classification model. The LinearSVM package implemented by scikit-learn[4] [18] is adopted to train the classifier. We randomly sample a portion of labeled nodes as the training data. Then we use the remainder nodes as the testing data. The proportion of training samples, denoted as T_r, varies from 10% to 50% for Citeseer and Google+ datasets, and from 1% to 5% for DBLP dataset. We run the experiment 10 times and average the results. The average accuracy is reported in Table 3.

From Table 3 one can see that the performances of MVC-DNE_{DB} and MVC-DNE_{EB} achieve similar performance, and both methods

[4]http://scikit-learn.org/

Table 3: Multi-class classification results (Accuracy) on the three datasets.

Model	Citeseer					Google+					DBLP				
T_r	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	1%	2%	3%	4%	5%
DeepWalk	0.309	0.339	0.354	0.363	0.367	0.546	0.551	0.554	0.552	0.554	0.619	0.646	0.660	0.667	0.674
LINE1st	0.460	0.507	0.525	0.538	0.547	0.476	0.488	0.492	0.493	0.497	0.588	0.612	0.624	0.630	0.635
LINE2nd	0.459	0.494	0.512	0.523	0.526	0.479	0.481	0.482	0.482	0.484	0.593	0.601	0.606	0.608	0.611
TADW	0.661	0.691	0.703	0.711	0.719	0.405	0.426	0.438	0.444	0.450	0.682	0.704	0.715	0.722	0.728
SDNE	0.515	0.544	0.556	0.566	0.574	0.569	0.575	0.576	0.577	0.579	0.659	0.678	0.691	0.699	0.705
SV-DNE$_P$	0.668	0.684	0.695	0.697	0.703	0.367	0.373	0.376	0.378	0.379	0.704	0.719	0.728	0.732	0.735
SDNE+SV-DNE$_P$	0.685	0.708	0.718	0.724	0.730	0.580	0.587	0.593	0.595	0.597	0.738	0.768	0.777	0.785	0.789
MVC-DNE$_{DB}$	0.722	0.733	0.740	0.743	0.744	**0.598**	**0.601**	**0.604**	**0.605**	**0.607**	**0.788**	**0.808**	**0.812**	**0.817**	**0.821**
MVC-DNE$_{EB}$	**0.726**	**0.739**	**0.746**	**0.752**	**0.755**	0.587	0.594	0.600	0.602	0.603	0.770	0.788	0.797	0.802	0.805



(a) Sparsity Sensitivity of Network Structure

(b) Sparsity Sensitivity of Nodes Properties

Figure 3: Embedding performance for original nodes w.r.t the proportion of removed links and nodes properties.

Table 4: Link prediction results (AUC) on the three datasets

Model	Citeseer	Google+	DBLP
DeepWalk	0.606	0.768	0.889
LINE$_{1st}$	0.663	0.804	0.901
LINE$_{2nd}$	0.655	0.733	0.715
TADW	0.923	0.522	0.843
SDNE	0.693	0.852	0.923
SV-DNE$_P$	0.884	0.463	0.825
SDNE+SV-DNE$_P$	0.905	0.731	0.945
MVC-DNE$_{DB}$	0.934	0.884	**0.960**
MVC-DNE$_{EB}$	**0.941**	**0.917**	0.938

outperform the baselines in all the cases. The proposed two methods significantly outperform DeepWalk, LINE and SDNE. One can also see that MVC-DNE$_{DB}$ and MVC-DNE$_{EB}$ are both consistently better than SDNE+SV-DNE$_P$, demonstrating that compared to a naive combination, our models can much more effectively integrate the complementary information of different views. On dataset Citeseer and DBLP, SV-DNE$_P$ outperforms DeepWalk and Line, which shows that the properties of nodes are essential to mine the network especially when the structure information is sparse. TADW does not achieve satisfactory performance comparing to deep methods. This indicates its performance largely depends on sufficient nodes properties. Due to the sparsity of network structure, TADW and SV-NDE$_P$ are comparable on Citeseer and DBLP datasets. The improvement on Google+ is less significant. This is mainly because Google+ is a much denser network with sufficient connections among nodes.

Performance evaluation on link prediction. To conduct link prediction, we randomly removed 50% edges from the original network and consider them as the links for prediction. Thus, the removed edges in the above mentioned operation are considered as positive samples. We then randomly select the same number of unconnected node pairs as negative links. The positive and negative node pairs form a balanced dataset of this task. We first compute the representation vectors of the original nodes with various embedding methods. Then we utilize the representation vectors to calculate the cosine similarity of each node pair. Area Under Curve (AUC) [6] is used as the performance evaluation metric, which can find the optimal threshold to predict positive and negative links

automatically. Thus AUC well reflects the correspondence of link labels and similarity scores.

Table 4 shows the experiment results. From this table, one can observe that our proposed two models are consistently better than the baseline methods. On Google+ dataset, LINE$_{1st}$ and SNDE outperform TADW and SDNE+SV-DNE$_P$ as the node properties in Google+ is rather sparse and noisy. Bringing in the noisy and sparse nodes properties may not be that helpful to network embedding learning. MVC-DNE$_{DB}$ and MVC-DNE$_{EB}$ still work well for effectively utilizing the view of the topology structure as the major information, and considering the node property view as complementary information.

MVC-DNE$_{EB}$ beats the best baseline SDNE+SV-DNE$_P$ on AUC in link prediction by around 10% on average on the all three datasets, demonstrating its power in deeply fusing data of the two views. The less desirable performances of MVC-DNE$_{DB}$ on Google+ demonstrate that the node properties do not help much in learning a better node representation as the node properties in Google+ is rather sparse and noisy. MVC-DNE$_{EB}$ performs much better than MVC-DNE$_{DB}$, because it learns a better node representation by more effectively utilizing the view of the topology structure as the major information, and only taking the node property view as complementary information.

5.5 Experiment Results on Incomplete Network

In this subsection, we conduct experiments to evaluate the performance of our proposed models on embedding incomplete networks. We will first describe how to build incomplete graphs from the views of network structure and nodes properties, respectively. Then we report and analyze the results.

Experiment on Incomplete Networks with Missing Links. To construct an incomplete network with missing links, we remove a proportion of links from the original networks. Thus the topology structure can be considered as incomplete. Specifically, the proportion of removed links increases from 0% to100% to make the structure information sparser and sparser. We then adopt our proposed models and 6 baselines to learn the corresponding embeddings on such incomplete networks.

We also evaluate the performance of various methods on the incomplete networks by conducting multi-class classification and link-prediction tasks. Figure 3(a) shows the details of results. One can see that with the increase of removed link proportion, the curves of classification accuracy and link prediction AUC score decrease. However, the decline trends of TADW, SDNE+SV-DNE$_P$ and our proposed MVC-DNE$_{DB}$, MV-DNE$_{EB}$ methods are significantly smaller than other methods, and they tend to be stable finally. It demonstrates the effectiveness of incorporating the node properties for network embedding on incomplete network with sparse links among nodes. Overall, MVC-DNE$_{DB}$ and MV-DNE$_{EB}$ are less sensitive with the increase of missing link proportions due to their power in multi-view correlation learning to alleviate the data sparsity issue in the network structure view.

Experiment on Incomplete Networks with Missing Node Properties. The features in the nodes property view can also be incomplete. Thus we randomly remove a proportion of properties of each node to build an incomplete network with missing nodes properties. After obtaining the representation vector of each node, we conduct multi-class classification to evaluate the performance. In this experiment, we do not compare with those baselines that can only handle topology structure information. The result is reported in Figure 3(b). One can see that the accuracies of all methods decrease quickly with the increase of proportion of the node properties. MVC-DNE$_{DB}$ and MV-DNE$_{EB}$ consistently outperform baseline methods SDNE and SV-DNE$_P$ as their accuracy curves are always above the curves of the other two methods.

5.6 Experiment Results for New Nodes Embedding

New nodes can be considered as an extreme case for incomplete networks. Previous models cannot effectively deal with the new nodes except for retraining the model, which is very time-consuming. In this section, we will first introduce how to prepare dataset, and then show the experiment results. Finally, we will explore and analyze the sensitivity of the models to the sparsity of the new nodes connections. Note that the baseline methods for new nodes embedding are SDNE, SV-DNE$_P$ and SDNE+SV-DNE$_P$.

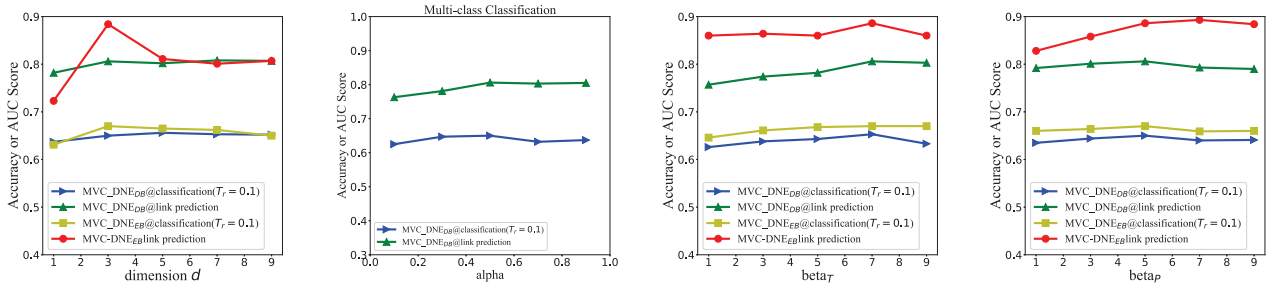
Data partition for new nodes. To construct news nodes for a network, we split each dataset into two parts by 9:1. 90% of the nodes are used as the original nodes and the remainder 10% nodes are considered as new nodes. Therefore, the adjacent matrix of the 90% original nodes are used as the structural input of models. By removing the 10% nodes, all the models are trained on the remaining 90% nodes. We assume that when a new node comes, there is very few connections between it and other nodes, and note that the new nodes also have sparse node property information. To let the two view data of the new nodes sparse, we randomly drop out 50% of the edges and properties for the new nodes.

Performance evaluation on multi-class classification. Similar to our pervious experiment, we use a proportion of labeled original nodes to train the LinearSVM. Note that T_r in this experiment denotes the portion of original nodes. The new nodes are considered as the test data. We report the classification accuracy of new nodes in Table 5. One can see that the performances of MVC-DNE$_{DB}$ and MVC-DNE$_{EB}$ are similar and both methods outperform all the baselines. The significant improvement over SDNE shows the effectiveness of our models in incorporating information from the nodes properties by multi-view learning.

Performance evaluation on link prediction. Differing from the experiment for original nodes embedding, we want to predict the edges connecting new nodes to the original nodes. To this aim, the removed edges in above mentioned drop-out operation are considered as positive samples. We then randomly select the same number of unconnected node pairs as negative links. Based on the map function f learned in the training step ,we firstly calculate the representation vectors of the new nodes with various embedding methods. After obtaining the cosine similarity of each node pair, we can adopt AUC to evaluate the performance of new nodes embedding on link prediction. Table 6 shows the experiment result. One can see that MVC-DNE$_{EB}$ and MVC-DNE$_{DB}$ outperform all the baselines. Specially, MVC-DNE$_{EB}$ outperforms the best baseline

Table 5: Multi-class classification results (Accuracy) on three datasets for new nodes.

Model	Citeseer					Google+					DBLP				
T_r	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	1%	2%	3%	4%	5%
SDNE	0.385	0.418	0.435	0.440	0.444	0.473	0.474	0.475	0.476	0.477	0.580	0.575	0.596	0.600	0.592
SV-DNE$_P$	0.612	0.623	0.628	0.632	0.640	0.371	0.378	0.379	0.381	0.380	0.641	0.652	0.660	0.666	0.665
SDNE+SV-DNE$_P$	0.634	0.653	0.663	0.663	0.670	0.481	0.484	0.485	0.489	0.490	0.663	0.687	0.7000	0.702	0.707
MVC-DNE$_{DB}$	0.650	0.665	0.667	0.674	0.675	0.505	0.510	**0.512**	**0.514**	0.515	**0.710**	**0.725**	**0.729**	**0.734**	**0.736**
MVC-DNE$_{EB}$	**0.670**	**0.680**	**0.681**	**0.682**	**0.683**	**0.509**	**0.511**	**0.512**	0.513	**0.517**	0.696	0.715	0.720	0.724	0.725



(a) dimension d (b) α (c) β_T (d) β_P

Figure 4: Embedding Performance for new nodes w.r.t parameter dimension d, hyper-parameter α, β_T and β_P.

Table 6: Link prediction results (AUC) on three datasets for new nodes

Model	Citeseer	Google+	DBLP
SDNE	0.721	0.697	0.831
SV-DNE$_P$	0.789	0.457	0.734
SDNE+SV-DNE$_P$	0.792	0.538	0.854
MVC-DNE$_{DB}$	0.806	0.730	0.895
MVC-DNE$_{EB}$	**0.884**	**0.750**	**0.896**

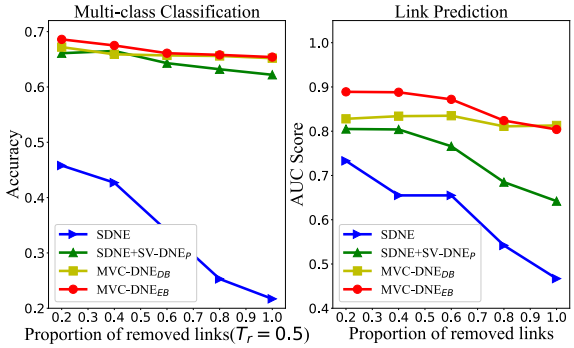


Figure 5: Embedding performance for new nodes w.r.t different proportions of removed links on Citeseer dataset.

SDNE+SV-DNE$_P$ by around 5% on average. One can also see that MVC-DNE$_{EB}$ slightly outperforms MVC-DNE$_{DB}$, because MVC-DNE$_{EB}$ is able to learn a characteristic representations in both views. The characteristics of network structure view is more useful for link prediction for new nodes when two views of data are both sparse.

Sparsity sensitivity analysis for new nodes embedding. We tune the proportion of removed links connecting the new nodes to the regular nodes from 20% to 100%. Then we evaluate the performance of the models on new nodes embeddings over Citeseer dataset. Figure 5 shows the results. One can see that 1) MVC-DNE$_{EB}$, MVC-DNE$_{DB}$, and SDNE+SV-DNE$_P$ perform significantly and consistently better than SDNE; and 2) the performances of all the methods drop with the increase of the removed link proportions. The dropping trend of SDNE is much more significant than the other three methods that consider both view information. It implies that the proposed methods are more robust to learn representations for new nodes whose links are very sparse due to their powerful ability to encode the correlated knowledge from the node properties.

5.7 Parameter Sensitivity Study

Finally, we study the sensitivity of the proposed models on the parameter represent dimension d, and hyper-parameters α, β_T and β_P. We use the classification accuracy and link prediction AUC score on dataset Citeseer for new nodes to evaluate the performance. The results are shown in Figure 4. Generally, it is important to set the number of dimensions for the latent embedding space for previous methods, but our method is not very sensitive to this parameter. α is only used in MVC-DNE$_{DB}$ to balance the weight between self-view reconstruction errors and cross-view reconstruction errors. As

shown in Figure 4(b), we find 0.5 is a reasonable choice. Figure 4(c) and 4(d) show that we should pay more attention on the non-zero elements in both views. β_T and β_P determine the reconstruction weight of non-zero elements in view T and P. The figures show that overall the performance of our models are not very sensitively to these parameters, demonstrating the robustness of the models.

6 CONCLUSION

In this paper, we propose a deep learning-based method MVC-DNE to perform network embedding with multi-view data. Specifically, we consider the topology structure and the node properties as two correlated views of a node in a network. Then we implement two models to learn the correlations between two views with the self-view and cross-view learning. Benefiting from the consistent information in multiple views, the learned node representations are much more robust to the incomplete networks. Experimental results on three real network datasets demonstrate the significant effectiveness of MVC-DNE, especially for efficiently generating representations for the new nodes.

7 ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grand Nos. U1636211, 61672081, 61602237, 61370126), National High Technology Research and Development Program of China (No.2015AA016004), fund of the State Key Laboratory of Software Development Environment (No. SKLSDE-2017ZX-19), and the Directors Project Fund of Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education (Grant No. 2017KF03).

REFERENCES

[1] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *International Conference on Machine Learning*. 1247–1255.
[2] Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* 15, 6 (2003), 1373–1396.
[3] Daniel Benyamin, Michael C McGinley, Michael Aaron Hall, and Nicholas J Bina. 2009. Social advertisement network. (May 18 2009). US Patent App. 12/467,981.
[4] Jianping Cao, Senzhang Wang, Fengcai Qiao, Hui Wang, Feiyue Wang, and S Yu Philip. 2016. User-guided large attributed graph clustering with multiple sparse annotations. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 127–138.
[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 891–900.
[6] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.
[7] Fangxiang Feng, Xiaojie Wang, and Ruifan Li. 2014. Cross-modal retrieval with correspondence autoencoder. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 7–16.
[8] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering* 19, 3 (2007).
[9] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation* 12, 10 (2000), 2451–2471.
[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.
[11] Qingbo Hu, Sihong Xie, Shuyang Lin, Senzhang Wang, and S Yu Philip. 2016. Clustering Embedded Approaches for Efficient Information Network Inference. *Data Science and Engineering* 1, 1 (2016), 29–40.
[12] Jure Leskovec and Julian J Mcauley. 2012. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*. 539–547.
[13] Chaozhuo Li, Senzhang Wang, Dejian Yang, Zhoujun Li, Yang Yang, and Xiaoming Zhang. 2017. PPNE: Property Preserving Network Embedding. In *DASFAA*. Springer, 163–179.
[14] Chaozhuo Li, Senzhang Wang, Dejian Yang, Zhoujun Li, Yang Yang, and Xiaoming Zhang. 2017. Semi-Supervised Network Embedding. In *DASFAA*. Springer, 131–147.
[15] Ping Li, Trevor J Hastie, and Kenneth W Church. 2006. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 287–296.
[16] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology* 58, 7 (2007), 1019–1031.
[17] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 689–696.
[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.
[20] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290, 5500 (2000), 2323–2326.
[21] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.
[22] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93.
[23] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 1067–1077.
[24] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 990–998.
[25] Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290, 5500 (2000), 2319–2323.
[26] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. 2016. Max-margin DeepWalk: discriminative learning of network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*. 3889–3895.
[27] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1225–1234.
[28] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 203–209. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14589
[29] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network Representation Learning with Rich Text Information.. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. 2111–2117.