Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Lifelong representation learning in dynamic attributed networks

Hao Wei, Guyu Hu, Wei Bai, Shiming Xia, Zhisong Pan*

Institute of Command and Control Engineering, Army Engineering University of PLA, Nanjing 210007, China

ARTICLE INFO

Article history:
Received 16 October 2018
Revised 9 May 2019
Accepted 14 May 2019
Available online 16 May 2019

Communicated by Dr Iosifidis Alexandros

Keywords:
Dynamic networks
Attributed networks
Lifelong learning
Network embeddings

ABSTRACT

Network embedding or network representation learning aims at learning a low-dimensional vector for each node in a network. The learned embeddings could advance various learning tasks in the network analysis area. Most existing embedding methods focus on plain and static networks while ignoring network dynamics. However, in real world networks, structure often evolves over time. In addition, many networks contain rich attributes and their attributes are changing over time. Naively applying existing embedding algorithms to each snapshot of dynamic network independently usually leads to unsatisfactory performance. In this paper, we present a Lifelong Dynamic Attributed Network Embedding Framework – LDANE. LDANE has a good mechanism to automatically expand the deep neural networks with the sizes of network growing and preserving what have learned from previous time steps. Furthermore, the proposed framework is carefully designed to flexibly incorporate the topology and attribute of nodes. We perform extensive experiments on both synthetic and real attributed networks to corroborate the significant advantages of the proposed framework compared with the existing state-of-the-art embedding techniques.

1. Introduction

Attributed networks are common in real-world systems such as social networks and scientific collaboration networks. For example, users in social networks may have profiles like gender, age and address. Researchers in scientific collaboration networks may have different research interests and foci. Social science theories [1,2] suggest that node attributes and network structure are highly correlated with each other. Modeling the correlation between node attributes and structure have already been applied in many fields such as sentiment analysis [3] and trust prediction [4], which have shown that attributes can be incorporated as the complementary content to enhance the performance.

Network embedding has recently attracted much attention [5,6]. The main idea of network embedding is to distill the high-dimensional information about a node to a dense vector embedding and preserve the node proximity in the embedded space. The vector can then be used to deal with tasks such as node classification, link prediction, clustering and social recommendation [7–9]. However, most of previous works have been designed only for plain networks and ignore the node attributes [5,6,10,11]. When the network is high sparsity, attributes can be the very useful complementary content to help learn better representations. In addition, in practical applications, many networks are dynamic and evolve over time with the addition, changing and deletion of nodes, edges and attributes. For example, in social networks, when people make new friends, the new edges will be created. People' attributes such as address, interests and jobs also change over time. The dynamic information in networks has been proved to be very important to understand networks [12,13]. Therefore, it is a critical requirement that the network embedding should reflect the evolution patterns of vertices well.

However, existing works often apply static embedding algorithms to each snapshot of the dynamic network. These methods do not perform well in terms of stability and scalability. The embeddings learned from static methods are always very different between consecutive time steps even though the networks do not change much. Meanwhile, learning embeddings independently for each snapshot is computationally expensive. Besides, modeling node's attributes information into dynamic networks is also a daunting task. Thus, dynamic attributed network embedding requires algorithm can combine attributes and topology well, but also to be an efficient online algorithm that can give embedding representations promptly.

To address above mentioned problems, in this paper, we develop a novel lifelong dynamic attributed network embedding framework – LDANE based on the lifelong learning DEN [14]. DEN can dynamically decide its network capacity to learn a compact overlapping knowledge sharing structure among tasks, which is very suitable for dynamic network embedding in deep neural

* Corresponding author.
E-mail address: 48220706@qq.com (Z. Pan).

https://doi.org/10.1016/j.neucom.2019.05.038
0925-2312/© 2019 Elsevier B.V. All rights reserved.

networks by regarding the dynamic network snapshots as a sequence of related tasks. In addition, we extract a set of attribute constraints according to the attributes of nodes and propose an objective function to restrict the learned embeddings to satisfy the attribute constraints. The attribute constraints can be updated very efficiently when the node attributes change, which is ideal for online learning dynamic attributed networks representation.

We summarize the main contributions of this paper as follows:

- We propose a novel dynamic attributed network embedding framework, which has a good mechanism to automatically expand the deep neural networks with the sizes of network growing and can preserve what have learned from previous time steps.
- We flexibly incorporate topology and attributes of nodes by extracting a set of constraints according to the attributes of nodes, which is ideal for online learning dynamic attributed networks representation.
- We perform extensive experiments on both synthetic and real-world attributed networks through graph reconstruction, link prediction and node classification. Experimental results show the superior performance of LDANE over state-of-the-art embedding methods.

The rest of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 states the problem of dynamic attributed network embedding. Section 4 presents the proposed framework LDANE. Experiments on synthetic and real datasets are presented in Section 5. Section 6 concludes the paper and visions the future work.

2. Related work

2.1. Network embedding

Most of earlier works mainly use matrix factorization techniques [15–19]. However, matrix factorization is too expensive to scale efficiently. More recently, motivated by skip-gram model [20], Deepwalk [21] and node2vec [22] sample an ordered sequence of nodes from a network by using random walks. These sequences can then be applied to skip-gram model to learn node embeddings. LINE [23] learns node embeddings by using first-order proximity and second-order proximity to preserve both the local and global structure information of networks. GraRep [24] extends LINE by considering high-order information. SDNE [25] is a deep model to capture the highly nonlinear structural information of networks. Recently, William et al. [26] proposed an inductive network embedding method GraphSAGE, which can be generalized to unseen nodes. Further, various approaches which are based on convolutional neural networks have been proposed to enhance the learned embeddings [27–29].

However, all the above approaches only work in plain networks. In real-word networks, there always exits attributes information of nodes. Zhu et al. [30] proposed a matrix factorization model to learn embeddings. The model combines both the links and node attributes. TADW [31] incorporates the text features of nodes into network embedding process, which is also a matrix factorization model. Huang et al. [32] uses the label information of nodes to help learn better nodes representation. GCN [33] learns node embeddings for attributed networks based on an efficient variant of convolutional neural networks. It requires the full graph Laplacian information during training. Nevertheless, these methods are all limited to deal with static networks.

In practical applications, many networks are dynamic and are continuously evolving over time. Hisano [34] proposed a simple discrete time semi-supervised graph embedding approach in dynamic networks. DynamicTriad [35] preserves both structural information and evolution patterns of a given network. DepthLGP [36] employs a deep neural network to learn a nonlinear transformation from latent states of the High-order Laplacian Gaussian Process to node embeddings. DynGEM [37] uses a dynamically expanding deep autoencoder to capture highly nonlinear first-order and second-order proximities of the network nodes. DyRep [38], an inductive deep representation learning framework that learns a set of functions to efficiently produce low-dimensional node embeddings that evolves over time. Nguyen et al. [39] described a general framework for learning time-respecting embeddings from continuous-time dynamic networks. However, all these methods are not considering the attributes of nodes. DANE [40] is a framework to tackle the problem of attributed network embedding in a dynamic environment. It leverages matrix perturbation theory to maintain the freshness of the end embedding results in an online manner.

2.2. Lifelong learning

Lifelong learning aims to leverage knowledge from earlier tasks to learn current task for obtaining better performance, or faster training speed on models for current task. It is often tackled as an online multi-task learning problem. ELLA [41] is a lifelong learning framework which is based on multi-task learning. It can update parameters of model efficiently based on previous tasks. EWC [42] regularizes the model parameter at each step which enables to find a good solution for catastrophic forgetting. In order to increase the capacity during training in denoizing autoencoder, Zhou et al. [43] add new neurons for the examples with high loss. Xiao et al. [44] propose an incrementally train network for multi-class classification, which extends the topmost layers of the network. DEN [14] is a novel deep network model that can be efficiently trained in an online manner. It is an effective incremental learning algorithm. By performing selective retraining, dynamic expansion, and splitting/duplicating neurons, it expands network capacity dynamically only the necessary number of units when a new task arrives, and effectively prevents semantic drift. In this paper, we consider the dynamic attributed network embedding under the lifelong learning scenario and propose a new framework based on DEN.

3. Problem definition

Attributed network G is defined as $G=(V,E,A)$, where $V=\{v_1,v_2,\dots,v_n\}$ denotes the node set, $E=\{e_{ij}\}$ is the edge set and $A\in\mathbb{R}^{n\times c}$ is the attribute matrix of nodes, here e_{ij} indicates an edge exists between v_i and v_j, c is the attribute dimensions. The weighted adjacency matrix of G is denoted by S. If $e_{ij}\in E$, $S_{ij}>0$; otherwise, $S_{ij}=0$. In addition, we use $S_i=[S_{i,1},\ \cdots S_{i,n}]$ to denote the ith row of the adjacency matrix.

Definition 3.1. Attributed network embedding. Given a network $G=(V,E,A)$, we aim to learn a matrix $Y\in\mathbb{R}^{|V|\times d}$, where d is the number of latent dimensions with $d\ll|V|$. The ith row of Y denotes the embedding vector of node i. The embedding should preserve node proximity both in topological structure and node attributes.

Node proximity in topological structure means the structural similarities between nodes, such as local structural similarities. Naturally, the local structures are represented by the observed links in the networks, and the learned embeddings of two connected nodes should also be similar. Node proximity in attributes means the attributes similarities between nodes. The embeddings of nodes with similar attributes should also be similar.

Definition 3.2. Dynamic attributed network. A series of dynamic attributed network snapshots is a set of networks $\mathcal{G}=$

Table 1
Notations for the framework.

Symbol	Definition
n	number of nodes in G
K	number of layers
$Y^{(k)} = \{y_1^{(k)}, \; y_n^{(k)}\}$	The k-th layer hidden representations
$Y = Y^{(K)}$	embedding
$W = \{W^{(k)}, \; \hat{W}^{(k)}, \; b^{(k)}, \; \hat{b}^{(k)}\}$	the overall parameters
$S = \{S_1, \ldots, \; S_n\}$	adjacency matrix of G
$\hat{S} = \{\hat{S}_1, \ldots, \; \hat{S}_n\}$	reconstructed data
neg_i	the most attribute dissimilar node set of node v_i
τ	The loss threshold
ρ_i^t	The semantic drift for unit i in autoencoder at time t
β	The semantic drift threshold

$\{G_1, \; \cdots, \; G_T\}$, where $G_t = (V_t, \; E_t, \; A_t)$ $(1 \le t \le T)$ and T is the number of snapshots.

In this paper, we consider three main components of dynamic attributed network that vary in time. First, nodes can join and disappear over time. In particular, we consider the disappeared nodes as part of the network with zero weights to the rest of the nodes. Second, links might be created and deleted throughout time. Finally, the attributes of nodes may change over time.

Definition 3.3. Dynamic attributed network embedding. Given a set of networks $\mathcal{G} = \{G_1, \; \cdots, \; G_T\}$, where $G_t = (V_t, \; E_t, \; A_t)$, dynamic attributed network embedding aims to learn a time-series of matrices $\mathcal{Y} = \{Y_1, \; \ldots, \; Y_t\}$ such that Y_t is the network embedding for G_t.

4. LDANE: lifelong dynamic attributed network embedding model

4.1. Framework

The frame of LDANE is shown in Fig. 1, and the terminology used is in Table 1. The symbols with hat on top are for the decoder. LDANE employs a deep autoencoder at its core. For each node v_i in the network, the autoencoder passes its neighborhood $S_i \in \mathbb{R}^n$ to learn its embedding $y_i = y_i^{(K)}$ at the outputs of the encoder, and the decoder reconstructs the neighborhood $\hat{S}_i$.

The top half of Fig. 1 shows the three components of the loss function of LDANE. The loss function of LDANE consists of L_{glob}, L_{loc} and L_{attr}. For any quad of nodes v_i, v_j, v_p and v_m, where v_j is the neighbor of v_i, $v_p \in pos_i$ is in the most similar node set of v_i in terms of attributes, and $v_m \in neg_i$ is in the most dissimilar node set of v_i in terms of attributes, L_{glob} is the reconstruction error of the output and input, L_{loc} is calculated from the embedding of node v_i and the embedding of its neighbor node v_j, L_{attr} is calculated from the embedding of node v_i, the embedding of node v_p and the embedding of node v_m. Section 4.2 describes the details of the loss function.

The lower half of Fig. 1 shows two snapshots of a dynamic network and the corresponding deep autoencoder at each step. When the next network snapshot comes, LDANE updates the deep autoencoder online by performing input and output expansion, selective retraining, autoencoder expansion and autoencoder split as needed. Section 4.3 describes the details of the update process.

4.2. Loss functions

SDNE [25] is the first to utilize autoencoder based deep network embedding model on static network. Here we also use a deep autoencoder to map the input data to a highly nonlinear latent space, and then reconstruct the input data. More specifically, for the node v_i with its neighborhood S_i, the hidden representation of each layer is defined as follows:

$$\begin{aligned} y_i^{(1)} &= \sigma\left(W^{(1)} S_i + b^{(1)}\right) \\ y_i^{(k)} &= \sigma\left(W^{(k)} y_i^{(k-1)} + b^{(k)}\right), \; k = 2, \ldots, K' \end{aligned} \tag{1}$$

where $\sigma(\cdot)$ represents the activation functions. After obtaining $y_i^{(K)}$, we can obtain the output $\hat{S}_i$ by reversing the calculation process of encoder. The goal is to minimize the following reconstruction error:

$$L_{glob} = \sum_{i=1}^{n} \left\| \hat{S}_i - S_i \right\|_2^2 \tag{2}$$

As we use the neighborhoods of nodes as the input to the autoencoder, i.e. S_i, the reconstruction process will make the nodes which have similar neighborhood structures have similar embeddings. In other words, the model can preserve the global network structure by reconstructing the neighborhood of nodes. However, networks in real world are always sparsity, the number of non-zero elements in S is far less than that of zero elements, which will be more prone to reconstruct the zero elements in S. To address this problem, the revised objective function is shown as follows:

$$L_{glob} = \sum_{i=1}^{n} \left\| \left(\hat{S}_i - S_i\right) \odot b_i \right\|_2^2 \tag{3}$$

where $\odot$ means the Hadamard product, b_i is a vector with $b_{ij} = 1$ if $S_{ij} = 0$ else $b_{ij} = \delta > 1$. This imposes more penalty to the reconstruction error of the non-zero elements than that of zero elements.

It is not enough to just preserve the global structure proximity, but also to preserve the local structure proximity. The local structure proximity means that nodes linked by an edge should be mapped near in the embedding space. The loss function for this goal is defined as follows:

$$L_{loc} = \sum_{i,j}^{n} S_{ij} \left\| y_i - y_j \right\|_2^2 \tag{4}$$

Inspired by the SWE [45], which incorporates the semantic knowledges into the word embedding process, we extract a set of attribute-related inequalities from the attribute similarity matrix $P \in \mathbb{R}^{|V| \times |V|}$ to model the rich attributes of nodes in the network. The attribute similarity matrix P is the attribute similarity scores between each pair of nodes. Users can design appropriate attribute similarity measurement based on the attributes and network characteristics, which guarantees the model can handle different types of attributes. According to the attribute similarity matrix P, we construct two nodes sets pos_i and neg_i for each node v_i in the network. The two sets both contain l nodes. The pos_i contains the top l similar nodes of node v_i and the neg_i contains the top l dissimilar nodes of node v_i, where $l \ll n/2$. For each node v_i, the following inequalities can be obtained:

$$P_{v_i v_k} > P_{v_i v_m}, \; v_p \in pos_i, \; v_m \in neg_i, \tag{5}$$

where $P_{v_i v_p}$ is the attribute similarity score between node v_i and node v_k. To retain the similarity of node attributes in node embeddings, the final embeddings of the corresponding nodes should satisfy the following constraints:

$$sim(y_i, \; y_p) > sim(y_i, \; y_m), \; v_p \in pos_i, \; v_m \in neg_i, \tag{6}$$

where $sim(y_i, \; y_p)$ is the cosine similarity between the embeddings of node v_i and node v_p in this paper. Based on the inequality constraints for each node in the network, we propose the following objective function to force the embeddings to satisfy the constraints:

$$L_{attr} = \sum_{v_i \in V} \sum_{v_m \in neg_i} \sum_{v_p \in pos_i} max(0, sim(y_i, y_m) - sim(y_i - y_p)). \tag{7}$$

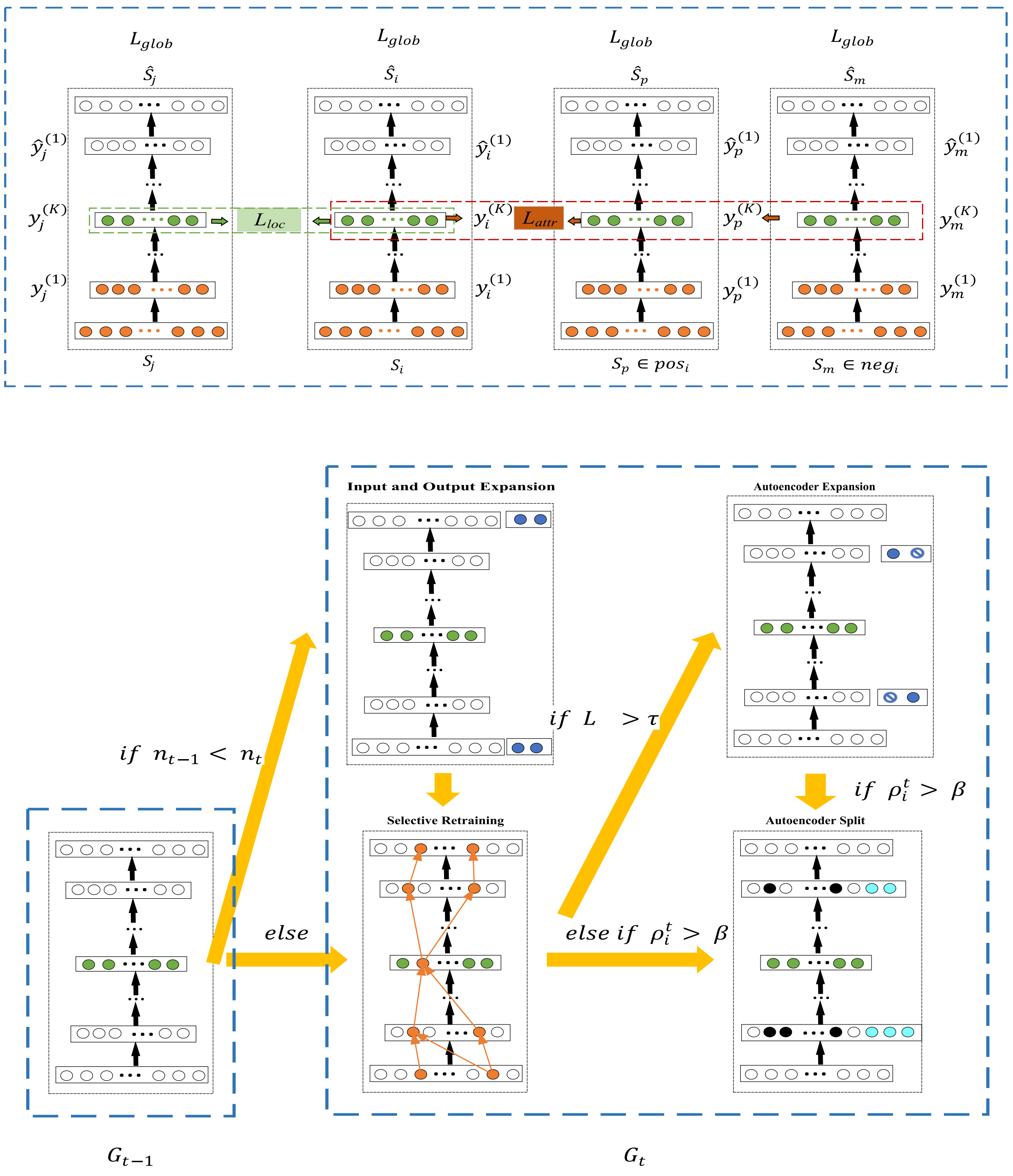


Fig. 1. LDANE: Dynamic Attributed Network Embedding Framework.

The purpose of the objective function is to ensure the embeddings similarity of two nodes with similar attributes should not be less than the nodes with dissimilar attributes. For each node in the network, we judge whether the current embeddings of the corresponding nodes satisfy the constraints. If not, we will update the embeddings towards the direction of satisfying these constraints.

To preserve the global structure proximity, local structure proximity and attribute similarity simultaneously, we combine Eqs. (3), (4) and (7) and joint minimizes the following objective function:

$$L = L_{glob} + \alpha_1 L_{loc} + \alpha_2 L_{attr} + \upsilon_1 L_1 + \upsilon_2 L_2, \quad (8)$$

where α_1, α_2, υ_1 and υ_2 are hyperparameters chosen as relative weights of the objective functions. Regularizers $L_1 = \sum_{k=1}^{K} (\|\boldsymbol{W}^{(k)}\|_1 + \|\hat{\boldsymbol{W}}^{(k)}\|_1)$ and $L_2 = \sum_{k=1}^{K} (\|\boldsymbol{W}^{(k)}\|_F^2 + \|\hat{\boldsymbol{W}}^{(k)}\|_F^2)$ are added to encourage sparsity in the network weights and prevent the model from overfitting the network respectively. In addition, we need the sparse connections in the next section to select retraining and expand the model.

4.3. Handling dynamic networks

Handling the dynamic network requires a good mechanism to automatically expand the autoencoder when the size of the network grows and preserve what have learned from previous time steps. We consider the problem under the lifelong learning scenario, where unknown number of network snapshots with unknown distributions of network data arrive at the model in sequence. Specifically, our goal is to learn models for a sequence of T snapshots of a dynamic network, $t = 1, \ldots, t, \ldots, T$. All the previous training datasets up to $t-1$ are not available at the current time t. Our autoencoder expansion method closely follows the work of Yoon et al. [14], but we have made many improvements to adapt to the dynamic network-structured data. It is very suitable for dealing with dynamic network embedding in autoencoder model. The lifelong learning agent at time t aims to learn the autoencoder parameter W^t by solving following problem:

$$\min_{W^t} L\left(W^t;\ W^{t-1},\ G_t\right),\ t = 1, \ldots \tag{9}$$

where L is the loss function defined in Eq. (8). Eq. (9) is an online manner to learn a compact overlapping knowledge sharing structure among network snapshots. We will describe our autoencoder expansion method in detail in following subsections.

Input and output expansion. The number of nodes in a dynamic network always changes over time. LDANE passes the neighborhoods of the nodes through the autoencoder to learn their embeddings. If the number of nodes at time t is smaller than the number of nodes at time $t-1$, we use zero-padded the input data, which means that a corresponding number of isolated nodes are added to the network. If the number of nodes at time t is greater than the number of nodes at time $t-1$, we must increase the number of input and output neurons of the autoencoder to meet the size of the input data. This will induce two parameter matrices expansions: $W_1^t = [W_1^{t-1};\ W_1^{\mathcal{N}}]$ and $W_H^t = [W_H^{t-1};\ W_H^{\mathcal{N}}]$, where $W^{\mathcal{N}}$ is the expanded weight matrix and H is the output layer. Then, we train only the input and output layer of weights with G_t by solving following problem:

$$\min_{W_h^{\mathcal{N}}} L\left(W_h^{\mathcal{N}};\ W_h^{t-1},\ G_t\right),\ h = 1, H \tag{10}$$

After expanding the input layer and the output layer, the model can adapt to new shape data. By optimizing Eq. (10), we pre-train the newly added weights and retain the other weights that are well-trained previously.

Selective retraining and expansion. Most of the previous works train the model for dynamic networks would be retraining the entire model for each snapshot of the dynamic network. However, such retraining will be very costly for a deep neural network. Thus, we would like to retrain only the weights that are affected by the new network snapshot and retain the other weights that are well-trained previously. To achieve this goal, when a new snapshot arrives, we first solve the following problem:

$$\min_{W_H^t} L\left(W_H^t;\ W_{1:H-1}^{t-1},\ G_t\right) \tag{11}$$

Notice that, we solve this optimization by only updating the topmost hidden units of the autoencoder. Since our loss function contains L_1 regularization, we will obtain the sparse connections between output units o_H^t and the hidden units at layer $H-1$. Once we build the sparse connection at this layer, we perform breadth-first search on the network starting from those selected nodes to identify all units that have paths to output units. Then we solve the following problem:

$$\min_{W_D^t} L'\left(W_D^t;\ W_D^{t-1},\ G_t\right) \tag{12}$$

where $L' = L_{glob} + \alpha_1 L_{loc} + \alpha_2 L_{attr} + \upsilon_2 L_2$ does not contains L_1 regularization since the sparse connections have been already established. W_D^t are the weights of the selected subnetwork. After selective retraining is done. The model checks if the loss meets the needs. If L is greater than the pre-defined threshold τ, we expand its capacity by k neurons at each hidden layer except the embedding layer K. If we expand the h_{th} layer of a network with k units, this will induce two parameter matrices expansions: $W_h^t = [W_h^{t-1};\ W_h^{\mathcal{N}}]$ and $W_{h-1}^{t-1} = [W_{h-1}^{t-1};\ W_{h-1}^{\mathcal{N}}]$ for outgoing and incoming layers respectively. Then we solve the following problem:

$$\min_{W_h^{\mathcal{N}}} L\left(W_h^{\mathcal{N}};\ W_h^{t-1},\ G_t\right) + \gamma \sum_g \left\| W_{h,g}^{\mathcal{N}} \right\|_2,\ \ h \neq 1,\ K,\ H \tag{13}$$

The second item in the Eq. (13) is the group sparsity regularization which is used in [46,47] to find the right number of neurons for a full network. Hidden units that are deemed unnecessary from the training will be dropped altogether. We use it to remove useless units in $W_h^{\mathcal{N}}$.

Autoencoder split. After we have obtained the network embeddings in new network snapshot, we need to evaluate the stability of the network embeddings over time. A successful dynamic network embedding method should be able to create a stable embedding over time. It means that if underlying networks change a little, the consecutive embeddings should have minor changes. To this end, we use L_2 regularization to regularize the parameters from deviating too much from its original values, which is shown as follows:

$$\min_{W^t} L\left(W^t;\ G_t\right) + \lambda \left\| W^t - W^{t-1} \right\|_2^2, \tag{14}$$

where λ is the regularization parameter. After performing Eq. (14), we calculate L_2 distance ρ_i^t between the weights at t and $t-1$ for each hidden unit i. If ρ_i^t is greater than the pre-defined threshold β, it means that the current features have significantly changed during training and then split i into two copies. After the duplication of all the needed units, the network needs to train the weights again by solving Eq. (14) since this operation have changed the network structure.

Attribute inequality constraints update. The attribute inequality constraints designed in this paper can be quickly updated according to the previous attribute information. At each network snapshot, we only focus on the nodes whose attributes changed compared with the previous snapshot. For example, at snapshot t, if the attributes of node i is different from $t-1$. We only need to update the $i\,th$ row and $i\,th$ column of the attribute similarity matrix P, and then update the pos and neg of every node. For each node j, if $P_{ji} > min(P_{jp})$, $p \in pos_j$ and $i \notin pos_j$, then node i replaces node p in pos_j. If $P_{ji} < max(P_{jq})$, $p \in neg_j$ and $i \notin neg$, then node i replaces node q in neg_j. In other cases, the pos_j and neg_j will remain unchanged. This can quickly update our attribute inequality constraints at each snapshot and is ideal for dynamic attributed network embedding.

Algorithm 1 describes our dynamically expandable network process.

5. Experiments

5.1. Datasets and baseline methods

Datasets. We evaluate the performance of our LDANE on two synthetic and two real-world dynamic networks. Table 2 shows the detailed information of the four datasets.

Synthetic Data (SYN): We generate two synthetic dynamic networks SYN-1 and SYN-2 using DANCer [48], which is a synthetic network generator for dynamic attributed networks with community structure that follows the properties of real-world networks.

DBLP: We extracted a DBLP co-author network from *dblp.org* for the authors that publish at least two papers between the years of 2007 and 2013 from seven different areas. The Bag-of-words model is applied on the paper title to get the attribute information. The major area the authors publish is regarded as ground truth.

Epinions[1]*:* which is a who-trust-whom online social network in which users share their reviews about products. Members of the site can also build trust networks to seek advice from others. Node attributes are formed by the bag-of-words model on the reviews, while the major categories of reviews by users are taken as the labels.

Baseline Methods. We compared the proposed method with several state-of-the-art methods using their published codes.

[1] http://www.epinions.com.

Algorithm 1 Dynamically Expandable Network.

```
Input: $G_t = (V_t, E_t, A_t)$, $G_{t-1} = (V_{t-1}, E_{t-1}, A_{t-1})$, autoencoder weights $W^{t-1}$, thresholds $\tau$, $\beta$
output: Embedding $Y_t^{(K)}$
1 From $G_t$, generate adjacency matrix $S$, penalty matrix $B$, attribute similarity matrix $P$
2 Update the inequality constraints of each node
3 Create the autoencoder model with initial architecture
4 Initialize $W^t$ randomly if $t = 1$, else $W^t = W^{t-1}$
5 if $|V_t| > |V_{t-1}|$ then
6    $W^t$ = Input and Output Expansion ($W^{t-1}$, $G_t$)
7 $W^t$ = Selective Retraining and Expansion ($W^{t-1}$, $G_t$, $\tau$)
8 $W^t$ = Network Split ($W^{t-1}$, $G_t$, $\beta$)
9 return the learned node embeddings $Y_t^{(K)}$

Input and Output Expansion
1 Input: $G_t$, $W^{t-1}$
2 Output: $W^t$
3 if $|V_t| > |V_{t-1}|$ then
4    Add $|V_t| - |V_{t-1}|$ units at input and output
5    Solve Eq. (10) to obtain $W^t$
6 return $W^t$

Selective Retraining and Expansion
1 Input: $G_t$, $W^{t-1}$,
2 Output: $W^t$
3 Initialize $D = \{o_H^t\}$
4 Solve Eq. (11) to obtain $W_H^t$
5 Add neuron $i$ to $D$ if $W_{Hi}^t \neq 0$
6 for $h = H - 1, \ldots, 1$ do
7    Add neuron $i$ to $D$ if there exists some neuron $j \in D$ such that $W_{h,ij}^t \neq 0$
8 Solve Eq. (12) to obtain $W_D^t$ and loss $L$
9 If $L > \tau$
10    Add $k$ units $W_h^{\mathcal{N}}$ at all hidden layer except the embedding layer $K$
11    Solve for Eq. (13)
12 for $h = H - 1, \ldots, 1$ do
13    Remove useless units in $W_h^{\mathcal{N}}$
14 return $W^t$

Network Split
1 Input: $G_t$, $W^{t-1}$, $\beta$
2 Output: $W^t$
3 Perform Eq. (14) to obtain $\hat{W}^t$
4 for all hidden unit $i$ except embedding layer $K$ do
5    $\rho_i^t = \|w_i^t - w_i^{t-1}\|_2$
6 If $\rho_i^t > \beta$ then
7    Copy $i$ into $i'$
8 Perform Eq. (14) with the initialization of $\hat{W}^t$ to obtain $W_H^t$
9 return $W^t$
```

Table 2
Detailed information of the datasets.

Data	Nodes	Links	Attributes	Categories	Time Steps
SYN-1	1000–1349	70,100–109,248	20	10	10
SYN-2	3000–3747	179,232–232,724	100	20	5
DBLP	6393–10,905	10,603–180,068	9835	9	7
Epinions	15,360–22,561	112,365–189,562	8536	20	10

- Deepwalk [21]: an algorithmic framework for learning feature representations for nodes in networks by word2vec and truncated random walk techniques.
- TADW [49]: is a shallow model incorporates rich text information of the network.
- SDNE [25]: is an autoencoder model to capture the highly nonlinear structural information of networks.
- DynGEM [37]: is a dynamically expanding deep autoencoder model to capture highly nonlinear first-order and second-order proximities of the graph nodes.

5.2. Results and analysis

In our experiments, we evaluate the performance on tasks of graph reconstruction, link prediction and node classification. For the graph reconstruction and link prediction, we use Mean Average Precision (MAP) [25] as our metric. For the node classification, we use the Micro-F1 and Macro-F1 as the results.

Table 3
Average MAP of network reconstruction.

	SYN-1	SYN-2	DBLP	Epinions
DeepWalk	0.084	0.024	0.27	0.092
TADW	0.431	0.207	0.46	0.23
SDNE	0.956	0.92	0.61	0.75
DynGEM	0.952	0.90	0.60	0.73
LDANE	**0.964**	**0.93**	**0.65**	**0.81**

Table 4
Average MAP of link prediction.

	SYN-1	SYN-2	DBLP	Epinions
DeepWalk	0.024	0.0786	0.042	0.021
TADW	0.205	0.0653	0.055	0.054
SDNE	0.298	0.180	0.0769	0.093
DynGEM	0.30	0.058	0.057	0.082
LDANE	**0.76**	**0.418**	**0.236**	**0.152**

5.2.1. Network reconstruction

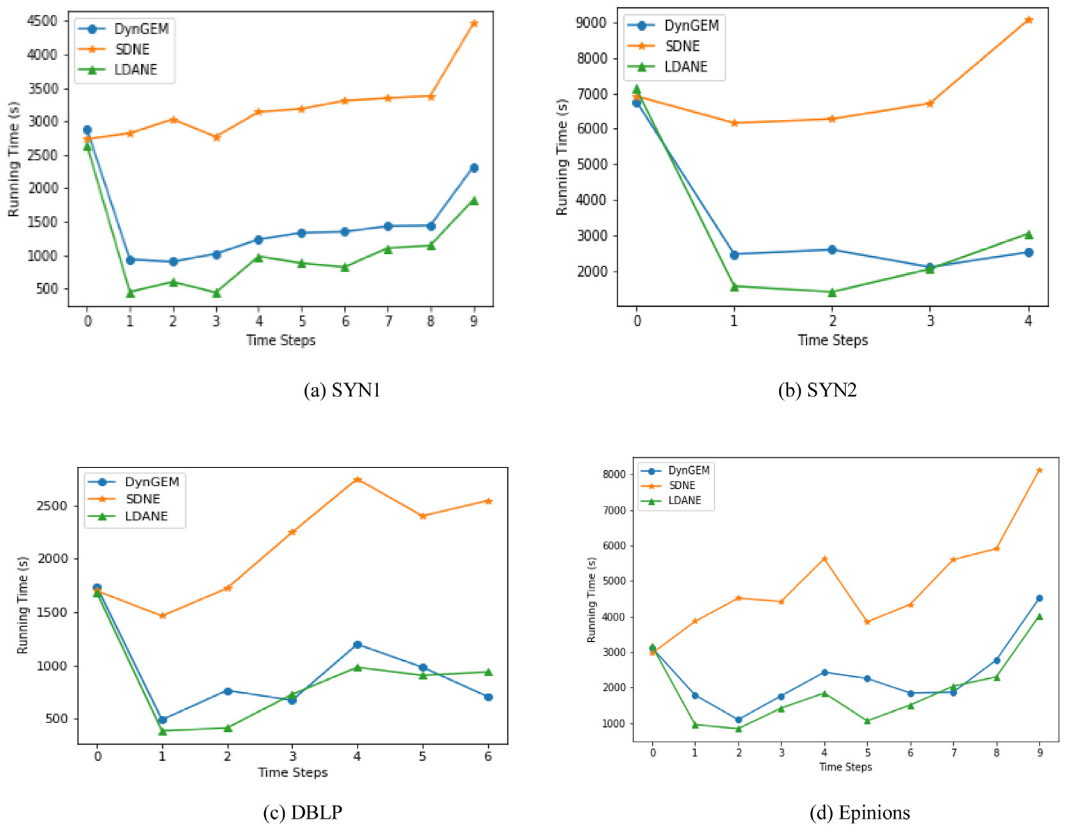
A good network embedding method should ensure that the learned embeddings can preserve the original network structure. As the existing links in the original network are known, we use the learned network representations to predict the links of the original networks and then evaluate the reconstruction performance. The network reconstruction MAP metric averaged over snapshots on our datasets are shown in Table 3. Table 3 shows that LDANE outperforms all the baselines, which demonstrates that LDANE is able to preserve the network structure well. LDANE, DynGEM and SDNE achieve better clustering performance than DeepWalk and TADW. The improvements indicate that the autoencoder model is effective in learning good node vectors for the task of network reconstruction.

5.2.2. Link prediction

In link prediction task, we randomly remove 15% of the edges at time t in the network which called E_{t_h}. We use the snapshots $\{G_1, \ldots G_{t-1}, G_t - E_{t_h}\}$ to learn a dynamic embedding and predict E_{t_h}. The most likely edges which are not in $G_t - E_{t_h}$ are regarded as the hidden edges. Then the precision scores are obtained by comparing the predictions against E_{t_h}. The prediction accuracy averaged over t from 1 to T is shown in Table 4. From Table 4 we can see that the representations learned by LDANE significantly outperforms the baselines, which demonstrates that LDANE is effective in linking prediction. Such an advantage is very important for some real-world applications such as recommendation and information retrieval.

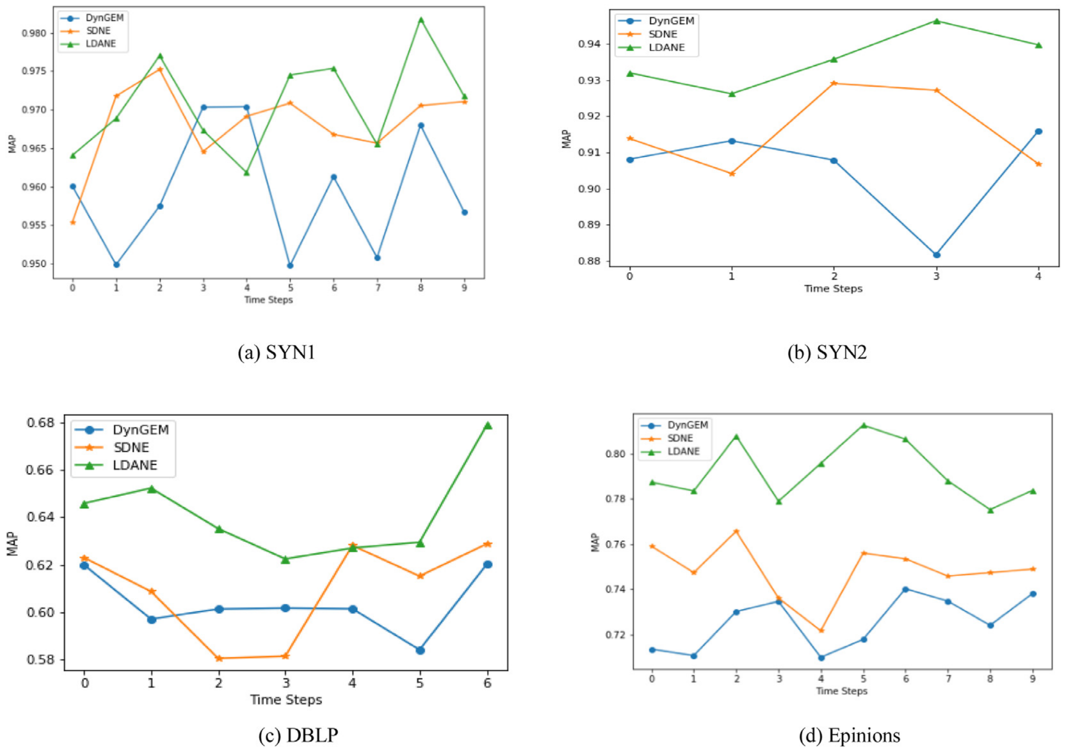
5.2.3. Multi-label classification

In real-world networks, only a subset of nodes in networks are labeled. Thus, node classification can be used to predict the labels of unlabeled nodes. In node classification task, the learned node embeddings are used as the input features of multi-class node classification model. We randomly sample a portion of labeled nodes as the training data, and the rest are the test data. The LinearSVM [50] package is adopted to train the classifier. We run the experiment 10 times and average the performance of both the Micro-F1 and Macro-F1 as the results. Table 5 shows the results. As we can see, in multi-label classification tasks, all baselines are doing well. In most cases, LDANE outperforms the baselines. It shows the effectiveness of our models to deal with node classification task.



(a) SYN1 (b) SYN2 (c) DBLP (d) Epinions

Fig. 2. Running time comparison.



(a) SYN1 (b) SYN2 (c) DBLP (d) Epinions

Fig. 3. Performance comparison.

Table 5
Multi-label classification results on the datasets.

	SYN-1		SYN-2		DBLP		Epinions	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
DeepWalk	0.94	0.92	0.93	0.92	0.73	0.71	0.15	0.13
TADW	**0.97**	0.92	0.82	0.77	0.71	0.65	0.21	0.19
SDNE	0.92	0.89	0.91	0.90	0.80	0.79	0.28	0.27
DynGEM	0.94	0.88	0.82	0.80	0.78	0.76	0.28	0.27
LDANE	0.95	**0.93**	**0.93**	**0.92**	**0.82**	**0.81**	**0.32**	**0.29**

5.2.4. *Efficiency*

We now compare efficiency of different embedding models. The comparison includes both performance and computation times in each network snap. As TADW is based on Graph Factorization and DeepWalk is based on Skip-gram model, although fast, the deep autoencoder based models are vastly outperformed compared with them. Therefore, we do not compare it with them. For the comparison of performance, due to the space limitation, we only post the result on network reconstruction task. Note that similar observations can be made on link prediction and node classification tasks. Fig. 2 shows the running time comparison results in each network snap. As can be observed from Fig. 2, the proposed LDANE is faster than other comparison methods in most network snaps. In the first network snap, since the three methods are all based on the deep autoencoder, the running time is similar. After that, the online methods LDANE and DynGEM are much faster than the offline method SDNE. Fig. 3 shows the performance comparison results. It is evident from the figure that LDANE beats other methods in most situations on network reconstruction task, which further verifies the effectiveness of our proposed LDANE.

6. Conclusions

Dynamic attributed networks are ubiquitous in real-world network and present new challenges for many learning problems because of their natural heterogeneity and dynamic. In this paper, we propose LDANE, a lifelong learning framework to construct embeddings for dynamic attributed networks. It has a good mechanism to automatically expand the deep autoencoder to capture highly nonlinear of the nodes. Furthermore, the propose framework is carefully designed to flexibly incorporates topology and attributes of nodes. We perform extensive experiments on both synthetic and real attributed networks to corroborate efficacy and efficiency of the proposed framework. Our future work will focus on how to extend the current framework to multi-dimensional dynamic networks and try to reduce the computation complexity of the proposed deep model.

Conflict of interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled, “Lifelong representation learning in dynamic attributed networks”.

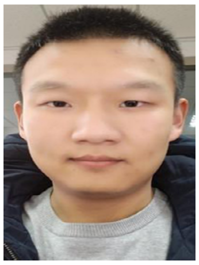
Acknowledgments

Our work is supported by the National Key Research Development Program of China (No. 2017YFB0802800). The authors would like to thank the Editor-in-Chief and anonymous reviewers for their insightful and constructive commendations that have led to an improved version of this paper.

References

[1] P.V. Marsden, Homogeneity in confiding relations, Soc. Netw. 10 (1) (1988) 57–76.
[2] M. McPherson, L. Smith-Lovin, J.M. Cook, Birds of a feather: Homophily in social networks, Annu. Rev. Sociol. 27 (1) (2001) 415–444.
[3] X. Hu, L. Tang, J. Tang, H. Liu, Exploiting social relations for sentiment analysis in microblogging, in: Proceedings of Conference on Web Search and Data Mining, WSDM, 2013.
[4] J. Tang, H. Gao, X. Hu, H. Liu, Exploiting homophily effect for trust prediction, in: Proceedings of Conference on Web Search and Data Mining, WSDM, 2013.
[5] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: Proceedings of Conference on Knowledge Discovery and Data Mining, KDD, 2014, pp. 701–710.
[6] J. Tang, Q. Meng, M. Wang, M. Zhang, J. Yan, Qiaozhu Mei, LINE: large-scale information network embedding, in: Proceedings of World Wide Web, WWW, 2015, pp. 1067–1077.
[7] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings of IEEE International Joint Conference on Neural Networks, 2, 2005, pp. 729–734.
[8] 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, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[9] N. Shervashidze, P. Schweitzer, E.J.v. Leeuwen, K. Mehlhorn, K.M. Borgwardt, Weisfeilerlehman graph kernels, J. Mach. Learn. Res. 12 (2011) 2539–2561.
[10] A. Grover, J. Leskovec, node2vec: scalable feature learning for networks, in: Proceedings of ACM SIGKDD International Conference, 2016.
[11] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of ACM SIGKDD International Conference, 2016.
[12] J. Neville, H. Goldberg, H. Goldberg, et al., Using relational knowledge discovery to prevent securities fraud, in: Proceedings of Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, ACM, 2005, pp. 449–458.
[13] M.E.J. Newman, The structure of scientific collaboration networks, Proc Natl Acad Sci USA 98 (2) (2001) 404–409.
[14] J. Yoon, E. Yang, J. Lee, et al. Lifelong Learning with Dynamically Expandable Networks. 2018.
[15] J.B. Tenenbaum, V.De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[16] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[17] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373–1396.
[18] P. Li, T.J Hastie, K.W. Church, Very sparse random projections, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 287–296.
[19] Q. Hu, S. Xie, S. Lin, S. Wang, S Yu Philip, Clustering embedded approaches for efficient information network inference, Data Sci. Eng. 1 (1) (2016) 29–40.
[20] W. Cheng, C. Greaves, M. Warren, From n-gram to skip-gram to concgram, Int. J. Corp. Linguist. 11 (4) (2006) 411–433.
[21] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 701–710.
[22] A. Grover, J. Leskovec, node2vec: scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 855–864.
[23] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: large-scale information network embedding, in: Proceedings of International Conference on World Wide Web, 2015, pp. 1067–1077.
[24] S. Cao, W. Lu, Q. Xu, Grarep: learning graph representations with global structural information, in: Proceedings of Conference on Knowledge Discovery and Data Mining, KDD, 2015.
[25] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1225–1234.
[26] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of NIPS, 2017.
[27] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: Proceedings of ICLR, 2014.
[28] T.N. Kipf, M. Welling. Variational Graph Auto-encoders. arXiv preprint arXiv:1611.07308, 2016.
[29] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of ICLR, 2017.
[30] S. Zhu, K. Yu, Y. Chi, Y. Gong, Combining content and link for classification using matrix factorization, in: Proceedings of SIGIR, 2007, pp. 487–494.
[31] C. Yang, Z. Liu, D. Zhao, M. Sun, E.Y. Chang, Network representation learning with rich text information, in: Proceedings of International Conference on Artificial Intelligence, 2015, pp. 2111–2117.
[32] X. Huang, J. Li, X. Hu, Label informed attributed network embedding, in: Proceedings of WSDM, ACM, 2017, pp. 731–739.

[33] T.N. Kipf, M. Welling, Variational graph auto-encoders, in: Proceedings of NIPS Workshop on Bayesian Deep Learning, 2016.
[34] R. Hisano, Semi-supervised graph embedding approach to dynamic link prediction, in: Proceedings of International Workshop on Complex Networks Springer, Cham, 2018, pp. 109–121.
[35] L. Zhou, Y Yang, X. Ren, F. Wu, Y. Zhuang, Dynamic network embedding by modelling triadic closure process, in: Proceedings of AAAI, 2018 2018.
[36] J. Ma, P. Cui, W. Zhu, DepthLGP: learning embeddings of out-of-sample nodes in dynamic networks, in: Proceedings of AAAI, 2018 2018.
[37] P. Goyal, N. Kamra, X. He, Y. Liu, Dyngem: deep embedding method for dynamic graphs, in: Proceedings of IJCAI International Workshop on Representation Learning for Graphs, 2017.
[38] Trivedi R., Farajtbar M., Biswal P., et al. Representation Learning over Dynamic Graphs 2018.
[39] G.H. Nguyen, J.B. Lee, R.A. Rossi, N.K. Ahmed, E. Koh, S. Kim, Continuous-time dynamic network embeddings, in: Proceedings of the 3rd International Workshop on Learning Representations for Big Networks, 2018.
[40] Li J., Dani H., Hu X., et al. Attributed Network Embedding for Learning in a Dynamic Environment 2017:387–396.
[41] E. Eaton, P.L. Ruvolo, ELLA: an efficient lifelong learning algorithm, in: S. Dasgupta, D. Mcallester (Eds.), Proceedings of JMLR Workshop and Conference, 28, ICML, 2013, pp. 507–515.
[42] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, in: Proceedings of the National Academy of Sciences, 2017.
[43] G. Zhou, K. Sohn, H. Lee, Online incremental feature learning with denoising autoencoders, in: Proceedings of International Conference on Artificial Intelligence and Statistics, 2012, pp. 1453–1461.
[44] T. Xiao, J. Zhang, K. Yang, Y. Peng, Z. Zhang, Error-driven incremental learning in deep convolutional neural network for large-scale image classification, in: Proceedings of the 22nd ACM International Conference on Multimedia, ACM, 2014, pp. 177–186.
[45] Q. Liu, H. Jiang, S. Wei, Z.-H. Ling, Y. Hu, Learning semantic word embeddings based on ordinal knowledge constraints, in: Proceedings of Association for Computational Linguistics, ACL, 2015, pp. 1501–1511.
[46] Wen W., Wu C., Wang Y., et al. Learning Structured Sparsity in Deep Neural Networks 2016.
[47] Alvarez J.M., Salzmann M. Learning the Number of Neurons in Deep Networks 2016.
[48] Benyahia O., Largeron C., Jeudy B., et al. DANCer: Ddynamic Attributed Network with Community Structure Generator 2016.
[49] C. Yang, Z. Liu, D. Zhao, M. Sun, E.Y. Chang, Network representation learning with rich text information, in: Proceedings of International Conference on Artificial Intelligence, 2015, pp. 2111–2117.
[50] 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, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.


Hao Wei received the M.S. degree in information and communication engineering from the PLA University of Science and Technology, in 2016. He is currently working toward the Ph.D. degree in Army Engineering University of PLA. His research interests include complex network in machine learning, network embedding, online time series prediction.

Guyu Hu received the B.S. degree in Radio Communication from Zhejiang University in 1983 (Hangzhou,China), M.S. degree in Computer Application Technology from Nanjing Institute of Communications in 1989 (Nanjing, China), and Ph.D. degree in Communications and Information Systems from Nanjing Institute of Communications in 1992. Since 1990, he devotes to the research on network management. From 1997, he has been a full professor in the PLA University of Science and Technology, China. From 1998, his research interests include intelligent of network management, mainly on failure-finding from data with pattern recognition, machine learning and neural networks.

Wei Bai, College of Command and Information System, Army Engineering University of PLA, Nanjing, China. Wei Bai was born in 1983 and received the M.S. from PLA University of Science and Technology, Nanjing, China, in 2008, at present doctor is reading. Since 2012, he has been an university lecturer and majored in network security, especially in security policy and security management.

Shiming Xia received the B.S. degree in radar engineering and M.S. degrees in information and communication engineering from PLA University of Science and Technology, Nanjing, China, in 2013 and 2016. He is currently pursuing the Ph.D. degree in computer science at the Army Engineering University of PLA, College of Command and Control Engineering, Nanjing, China. His research interest includes the network security, reinforcement learning.

Zhisong Pan received the B.S. degree in computer science and M.S. degree in computer science and application from the PLA Information Engineering University, Zhengzhou, China, in 1991 and 1994, respectively, and the Ph.D. degree in Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2003. From July 2006 to present, he led several key projects of intelligent data processing for the network management. From July 2011, he has working as a full professor in the PLA University of Science and Technology, China. His current research interests mainly include pattern recognition, machine learning and neural networks.