World Wide Web (2022) 25:239–258
https://doi.org/10.1007/s11280-021-00987-z

API-GNN: attribute preserving oriented interactive graph neural network

Yuchen Zhou[1,2] · Yanmin Shang[1,2] · Yanan Cao[1,2] · Qian Li[3] · Chuan Zhou[1,4] · Guandong Xu[3]

Received: 17 August 2021 / Revised: 18 November 2021 / Accepted: 8 December 2021 /
Published online: 23 January 2022

Abstract
Attributed graph embedding aims to learn node representation based on the graph topology and node attributes. The current mainstream GNN-based methods learn the representation of the target node by aggregating the attributes of its neighbor nodes. These methods still face two challenges: (1) In the neighborhood aggregation procedure, the attributes of each node would be propagated to its neighborhoods which may cause disturbance to the original attributes of the target node and cause over-smoothing in GNN iteration. (2) Because the representation of the target node is derived from the attributes and topology of its neighbors, the attributes and topological information of each neighbor have different effects on the representation of the target node. However, this different contribution has not been considered by the existing GNN-based methods. In this paper, we propose a novel GNN model named API-GNN (Attribute Preserving Oriented Interactive Graph Neural Network). API-GNN can not only reduce the disturbance of neighborhood aggregation to the original attribute of target node, but also explicitly model the different impacts of attribute and topology on node representation. We conduct experiments on six public real-world datasets to validate API-GNN on node classification and link prediction. Experimental results show that our model outperforms several strong baselines over various graph datasets on multiple graph analysis tasks.

Keywords Data mining · Graph neural networks · Social analysis · Representation learning

1 Introduction

Attributed network in which nodes are usually associated with rich attributes information is ubiquitous, such as social networks, citation networks, E-commerce networks and so on. Attributed network embedding, which aims to learn low-dimensional representations of nodes,is a basic task for attributed network analysis. However, how to use the network

✉ Yanmin Shang
shangyanmin@iie.ac.cn

Extended author information available on the last page of the article

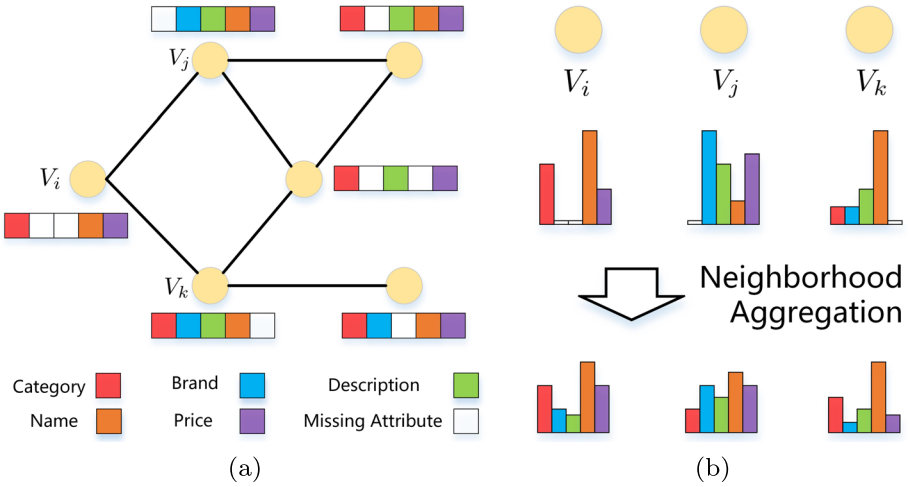


Fig. 1 Disturbance of GNNs neighborhood aggregation on node attribute. The vector of each node in subgraph (a) represents its initial attributes and subgraph (b) represents the attributes of nodes after GNNs neighborhood aggregation. After neighborhood aggregation, the missing attributes (white boxes) in some nodes' initial attributes are filled, while the existing attributes (column height in histogram) in some nodes' initial attributes are greatly changed. This change of attribute is called disturbance to the original attribute of node

topology information and node attribute information effectively to generate the meaningful node representation for various downstream tasks is a great challenge.

There has been many conventional graph embedding methods which encode topology structure information to learn node representation [4, 14, 15, 17, 21]. However, those methods ignore the attribute information which is also important to node representation. More recently, there has been a surge of graph neural networks (GNNs) whose core is neighborhood aggregation. They can combine both the topology structure information and node attribute information to learn meaningful node representation. Now, GNNs have shown great popularity in tackling various graph analytics problems for attributed networks such as node classification [22, 25, 28], link prediction [19, 23] and recommendation [8, 16].

However, there are still some drawbacks in most of existing GNNs. On one hand, iterative neighborhood aggregation in GNNs would disturb the attributes of nodes. Especially when the attributes of target node and the attributes of its neighbor node are extremely different, it will make the target node's attribute deviate significantly from the original one [26]. Furthermore, as the neighborhood aggregation smoothens the attributes, iterative aggregation may make all nodes have the identical representation which reduces the discriminability of nodes. We take a subgraph from the JD dataset for an example. As elaborated in Figure 1(a), node V_j and node V_k are the neighbors of node V_i. After several rounds of neighborhood aggregation in GNNs, we observed that: (1) the missing attributes (two white boxes) of node V_i have been filled with blue and green attributes; (2) the original orange attribute value of node V_j has been significantly changed as shown in Figure 1(b); (3) the attributes of node V_i and node V_k becoming more similar. On the other hand, neighborhood aggregation naturally fuses attribute and topology information, making them jointly affect the representation of the target node. This mechanism can't distinguish the contributions of the attribute and topology information. In fact, the impacts of neighbor node's attribute and topology on target node representation are different.

In order to improve the performances of GNNs in attributed graph embedding, some researchers have made a breakthrough in eliminating the consistency of node representations. CS-GNN [6] and ALaGCN [24] prevent representations of nodes from becoming identical by designing an adaptive neighborhood aggregation for each node, GResNet [27] keeps the diversity of node representations by exploring various residual connections and DAGNN [10] decouple the representation transformation and propagation in graph convolution operations for a deeper GNN. However, as far as we know, these methods haven't solve the node attribute disturbance problem and distinguish the impacts of attribute and topology on neighborhood aggregation. To address these two issues, we propose a novel graph neural network named *Attribute Preserving Oriented Interactive Graph Neural Network* (API-GNN). API-GNN consists of three modules: original attribute aggregation module (AGM), topology aggregation module (TAM) and interaction module (IM). AGM models the original attributes of neighbor nodes on the target node global representation and keeps the node original attribute representation unchanged in GNNs iterations. TAM distills the neighbor topology information from the a neighbor node global representation and models the topology information of neighbor nodes on the target node global representation. These two factors interact in the IM to jointly determine the global representation of the target one. In this way, API-GNN can not only weaken the disturbance of attributes, but also take into account the correlation and different effects of attribute and topology of neibourhood on node representation.

In general, the main contributions of our method are as follows:

- For the first time, we explore the problems of GNN methods in neighbor aggregation, that is, the original attribute disturbance problem, and the influence differences of attribute and topology information on node representation.
- We propose a novel API-GNN to solve the above problems. In the process of neighbor aggregation, it can not only ensure that the original attributes are preserved as much as possible, but also consider the differences and associations of attributes and topology to node representation through an interactive way.
- We conduct extensive experiments on node classification and link prediction tasks with six real-world attributed network datasets. The results demonstrate that API-GNN outperforms all baseline models.

2 Preliminary

In this section, we first introduce the notions of attributed networks and formally introduce the studied problem of attributed network embedding. Next, we give an intuitive explanation of our proposed attribute embedding, which is for better understanding of API-GNN. In this paper, we use calligraphic fonts, bold lowercase letters, and bold uppercase letters to denote sets (e.g., $\mathcal{G}$), vectors (e.g., $\mathbf{x}$), and matrices(e.g., $\mathbf{X}$), whereas others are scalars. For convenience, some important notations are summarized in Table 1.

2.1 Problem definition

We assume an attributed network can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, where $\mathcal{V} = \{V_1, V_2, ..., V_N\}$ is the node set ($N = |\mathcal{V}|$). $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the edge set between nodes in $\mathcal{V}$. $\mathbf{A} \in \{0, 1\}^{N \times N}$ denotes the adjacency matrix of $\mathcal{G}$, entry $a_{ij} = 1$ indicates that there is

Table 1 Table of main symbols

Symbols	Definitions
$\mathcal{G}$	Input attributed network
$\mathbf{A}$, $\mathbf{X}$, $\mathbf{Y}$	Matrices of adjacency, attribute and node label
N, M	The number of nodes and node attributes
$\mathcal{C}$	The labels set of nodes
$\mathcal{N}(i)$	The neighbor nodes set of node V_i
$\mathbf{x}_{it}$	t-th original attribute vector of node V_i
$\mathbf{M}_F^{(l)}, \mathbf{M}_T^{(l)}$	Layer-specific transformation matrix
$\mathbf{a}_F$, $\mathbf{a}_N$	Attribute-level and node-level attention vector
$\mathbf{W}_*$, $\mathbf{b}_*$	Model trainable parameters
$\mathbf{h}_i^{(l)}$	Global representation of node V_i in l-th layer
$\mathbf{of}_i^{(l)}, \mathbf{tp}_i^{(l)}$	Original attribute representation with low dimension and topology information representation of node V_i in l-th layer
$\mathbf{hf}_i^{(l)}, \mathbf{ht}_i^{(l)}$	Neighborhood attribute representation and neighborhood topology representation of node V_i in l-th layer
$\mathcal{NI}(i)$	Neighborhood information representation of node V_i in l-th layer
$\mathcal{Y}_\mathcal{C}, \mathcal{Y}_+, \mathcal{Y}_-$	Labeled nodes set, postive instances link set and negative instances link set
$\lambda_1, \lambda_2, \lambda_3$	Weight coefficients of loss function

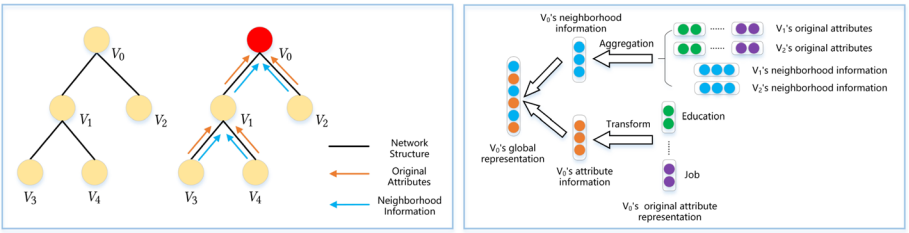
an edge between node V_i and node V_j, otherwise, $a_{ij} = 0$. $\mathbf{X} = [\mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_M}] \in \mathbb{R}^{N \times D}$ represents the matrix composed of all attribute vectors of all nodes (N and M represent the number of nodes and attributes respectively, D is all attributes dimension of $\mathbf{X}$), where each attribute of each node is represented as a vector. $\mathbf{X_i} = [\mathbf{x_{1i}}, \mathbf{x_{2i}}, ..., \mathbf{x_{Ni}}]^T \in \mathbb{R}^{N \times D_i}$ is the i-th attribute matrix of all nodes, where D_i is the i^{th} attribute dimension of $\mathbf{X_i}$ and $D = \sum_{i=1}^{M} D_i$. Specifically, in real-world, each node always has many attributes, for instance, user has name, gender, age and so on in social network, item has brand, price, category and so on in e-commerce network.

Definition 1 Attributed Network Embedding: Given an attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, Attributed Network Embedding aims to learn an embedding model $f : V \to \mathbb{R}^d$ that embed each node $V \in \mathcal{V}$ to a vector in a d-dimensional space $\mathbb{R}^d$ with A and X, $d \ll |\mathcal{V}|$. The node embedding can be applied to all kinds of downstream tasks.

In this paper, our goal is to make attribute network embedding meet the following **two conditions**: (1) the original attributes of nodes should be kept as much as possible in the embedding process. (2) The different influence of neighbor node's attribute and topology on the representation of target node should be modeled.

2.2 Our solution

In order to make attribute network embedding meet the above definition and conditions, we propose a new GNN aggregation mechanism which is shown in Figure 2. To better illustrate our aggregation mechanism, we first give some definitions.



(a) Message propagation of node V_0 (b) Aggregation of node V_0

Fig. 2 The aggregation method of API-GNN

Definition 2 Attribute Information: For node V_i ($V_i \in \mathcal{V}$), the representation of attribute information $\mathbf{of}_i$ is generated from the original attribute vector of V_i itself, i.e. $x_i = [x_{i1}, x_{i2}, ..., x_{iM}]$ ($x_i \subset X$), through linear or nonlinear transform.

Definition 3 Neighborhood Information: For node V_i ($V_i \in \mathcal{V}$), the neighborhood information $\mathcal{NI}(i)$ is the information propogated to it by its neighbors $\mathcal{N}(i)$, which includes the original attributes x_j of neighbor node V_j ($V_j \in \mathcal{N}(i)$) and the neighborhood information $\mathcal{NI}(j)$ propogated by neighbors of neighbor node V_j. So neighborhood information is generated by an iterative procedure.

As shown in Figure 2, each node independently propagates its original attributes to its neighbors. In our GNN aggregation procedure, each node's representation (global representation) is composed of two kinds of information, namely attribute information of the node itself (Definition 2) and neighborhood information (Definition 3). Taking node V_0 as an example (Figure 2(b)), its global representation is obtained by aggregating attribute information and neighborhood information of V_0. The neighborhood information of V_0 is obtained by aggregating the neighborhood information and original attributes of neighbors V_1 and V_2, and the neighborhood information of V_1 is obtained by aggregating the neighborhood information and original attributes of neighbors V_3 and V_4.

Definition 4 Topology Information: According to the iterative characteristics of neighborhood information, the essence of neighborhood information is that the original attributes propogation through hierarchical topology. Therefore, the neighborhood information can be obtained from the original attribute information and topology information. It should be noted that the topology information here is not the first-order neighbor relationship, it refers to the overall influence of hierarchical topology in message propagation, so it cannot be obtained directly.

Based on this intuitive idea, we innovatively propose an attributed embedding model named Attribute Preserving Oriented Interactive Graph Neural Network. Next, we will introduce our model in detail.

3 The proposed model

This section introduces our approach API-GNN in details. As shown in Figure 3, API-GNN contains three modules: the original attribute aggregation module, topology aggregation module and interaction module. The first two modules model the influence of the neighbors' attribute and topology on node representation independently, and obtain attribute representation and topology representation of the neighbourhood. The third module models the correlation between attribute representation and topology representation of neighbourhood on target node representation, and obtains target node global representation. We utilize original attribute aggregation module to eliminate the attribute disturbance caused by neighborhood aggregation and keep the original attribute of nodes as unchanged as possible.

3.1 Original attribute aggregation module

The purpose of the original attribute aggregation module is to model the different influences of the neighbor nodes' original attributes on the target node global representation in the GNN iteration process. The output of the module is the neighborhood attribute representation of target node. In GNN aggregation, the attributes of each neighbor has different influence on the target node neighborhood attribute representation. Moreover, in attributed network, each neighbor node has multiple attributes, and each attribute also has different influence on the target node neighborhood attribute representation. Based on the above, in this module, we design two levels of attention (attribute-level attention and node-level attention) to model these two effects separately.

3.1.1 Attribute-level attention

This step models the influences of different original attributes of a neighbor node on the target node global representation. We define the layer-specific attribute transformation

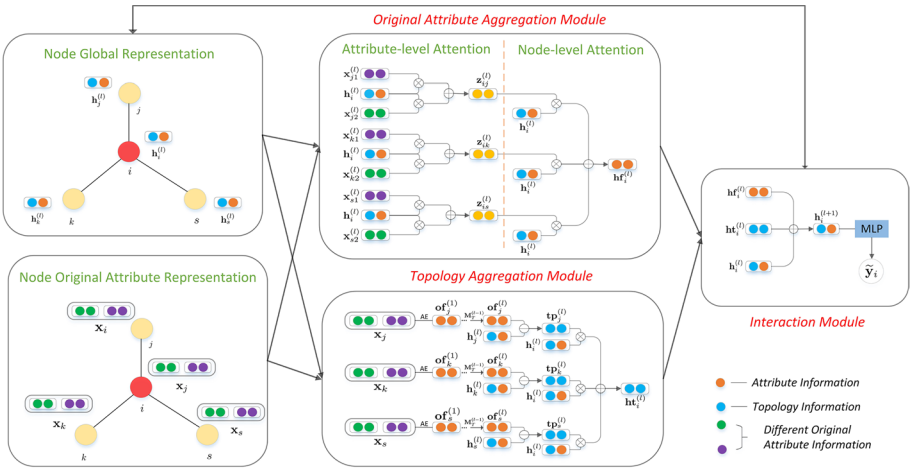


Fig. 3 The overall framework of the proposed API-GNN

matrix $\mathbf{M}_F^{(l)}$ to project the each original attribute vector of a neighbor node into the semantic space of the target node global representation. The projection process can be shown as:

$$\mathbf{x}_{jt}^{(l)} = \mathbf{M}_F^{(l)} \cdot \mathbf{x}_{jt} \tag{1}$$

we define node V_j is a neighbor node of node V_i, $\mathbf{x}_{jt}$ is the t-th original attribute vector of node V_j and $\mathbf{x}_{jt}^{(l)}$ is the projected attribute vector. After projecting, we leverage attention mechanism in GAT [20] to learn the weight between target node global representation in the l-th layer $\mathbf{h}_i^{(l)}$ and each projected attribute vector $\mathbf{x}_{jt}^{(l)}$. It can be formulated as follows:

$$e_{ijt}^{(l)} = LeakyReLU(\mathbf{a}_F^T \cdot [\mathbf{W}_1^{(l)}\mathbf{h}_i^{(l)} || \mathbf{W}_1^{(l)}\mathbf{x}_{jt}^{(l)}) \tag{2}$$

$$\alpha_{ijt}^{(l)} = softmax(e_{ijt}^{(l)}) = \frac{\exp(e_{ijt}^{(l)})}{\sum_{t=1}^{M} \exp(e_{ijt}^{(l)})} \tag{3}$$

where $\mathbf{a}_F$ is the attribute-level attention vector, $\mathbf{W}_1^{(l)}$ is attribute-level weight matrix in layer l, M is the attribute number of node and $\|$ denotes the concatenate operation. $\alpha_{ijt}^{(l)}$ is the attention weight which reflects the important of t-th attribute of node V_j for node V_i. The influence of all the original attributes of node V_j on the neighborhood attribute representation of target node V_i in the l-th layer is denoted as $\mathbf{z}_{ij}^{(l)}$, here σ is the activation function.

$$\mathbf{z}_{ij}^{(l)} = \sigma\left(\sum_{t=1}^{M} \alpha_{ijt}^{(l)} \cdot \mathbf{x}_{jt}^{(l)}\right) \tag{4}$$

3.1.2 Node-level attention

This step models the influences of different neighbors' attributes on the target node global representation. We adopt the similar attention mechanism similar to attribute-level attention.

$$e_{ij}^{(l)} = LeakyReLU(\mathbf{a}_N^T \cdot [\mathbf{W}_2^{(l)}\mathbf{h}_i^{(l)} || \mathbf{W}_2^{(l)}\mathbf{z}_{ij}^{(l)}]) \tag{5}$$

$$\alpha_{ij}^{(l)} = softmax(e_{ij}^{(l)}) = \frac{\exp(e_{ij}^{(l)})}{\sum_{j \in \mathcal{N}(i)} \exp(e_{ij}^{(l)})} \tag{6}$$

where $\mathbf{a}_N$ is the node-level attention vector, $\mathbf{W}_2^{(l)}$ is node-level trainable weight matrix and N_i is the neighbor set of node V_i. $\alpha_{ij}^{(l)}$ is the attention weight which indicates the importance of attributes of neighbor node V_j to target node V_i. We use above attention weights to acquire the target node neighborhood attribute representation:

$$\mathbf{hf}_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} \cdot \mathbf{z}_{ij}^{(l)}\right) \tag{7}$$

It should be noted that $\mathbf{hf}_i^{(l)}$ only depends on the aggregation of neighbor nodes' original attributes, and does not use the topology information of neighbors. Therefore, the disturbance of topology to the original attributes of node is avoided.

3.2 Topology aggregation module

The purpose of topology aggregation module is to model the influence of neighbor topology on target node global representation, and the output of this module is the neighborhood topology representation of the target node. This module includes two steps: Topology Distilling and Topology Aggregation.

3.2.1 Topology distilling

Due to the integration of topology information and attribute information in the iterative aggregation of existing GNNs, it is difficult to obtain pure topology information. In this paper, we use the node global representation minus node original attributes vector in each API-GNN layer to obtain the topology information. Because the node has many dimensional attributes, the node original attributes vector dimension is high, so we use a two layers auto-encoder to obtain low-dimensional original attributes vector in the first API-GNN layer, and make the dimension consistent with the node global representation in API-GNN iteration. The encoder of the auto-encoder can be represented as:

$$\mathbf{x}_i = \mathbf{x}_{i1}||\mathbf{x}_{i2}||\cdots||\mathbf{x}_{iM} \tag{8}$$

$$\mathbf{of}_i^{(1)} = \sigma(\mathbf{W}_4 \cdot \sigma(\mathbf{W}_3 \cdot \mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2) \tag{9}$$

where $\mathbf{x}_i$ is the concatenation of all original attribute vectors of node V_i, $\mathbf{W}_3, \mathbf{W}_4, \mathbf{b}_1, \mathbf{b}_2$ are the trainable parameters of encoder and $\mathbf{of}_i^{(1)}$ is the low-dimensional original attributes vector of the node V_i in the first API-GNN layer. The decoder and loss function of auto-encoder can be defined as follow:

$$\mathbf{x}_i' = \sigma(\mathbf{W}_6 \cdot \sigma(\mathbf{W}_5 \cdot \mathbf{of}_i^{(1)} + \mathbf{b}_3) + \mathbf{b}_4) \tag{10}$$

$$L_{auto} = \frac{1}{N}\sum\nolimits_{i=1}^{N} ||\mathbf{x}_i' - \mathbf{x}_i||^2 \tag{11}$$

By solving the above objective function, we can obtain $\mathbf{of}_i^{(1)}$ and use it to initialize the node global representation $\mathbf{h}_i^{(1)}$ in the first API-GNN layer. With the iteration of API-GNN, the global representation of node is constantly changing and the original attributes vector of node is also changing. We define a layer-specific transformation matrix $\mathbf{M}_T^{(l-1)}$, and it transforms the original attributes vector of a node in previous layer into original attributes vector in current layer.

$$\mathbf{of}_i^{(l)} = \mathbf{M}_T^{(l-1)} \cdot \mathbf{of}_i^{(l-1)} \tag{12}$$

After obtaining the original attributes vector of a node, we distill its topology information $\mathbf{tp}_i^{(l)}$ in the l-th layer as:

$$\mathbf{tp}_i^{(l)} = \mathbf{h}_i^{(l)} - \mathbf{of}_i^{(l)} \tag{13}$$

It should be noted that in the first API-GNN layer, we set $\mathbf{tp}_i^{(1)}$ to the zero vector.

3.2.2 Topology aggregation

Because the contribution of topology information of different neighbor nodes to the target node global representation is different, we define an attention mechanism to model the difference. It is similar to the attention mechanism in original attribute aggregation module. We formulate it as follow:

$$r_{ij}^{(l)} = LeakyReLU(\mathbf{a}_T^T \cdot [\mathbf{W}_7^{(l)}\mathbf{h}_i^{(l)} || \mathbf{W}_7^{(l)}\mathbf{tp}_j^{(l)}]) \tag{14}$$

$$\beta_{ij}^{(l)} = softmax(r_{ij}^{(l)}) = \frac{\exp(r_{ij}^{(l)})}{\sum_{j \in \mathcal{N}(i)} \exp(r_{ij}^{(l)})} \tag{15}$$

$$\mathbf{ht}_i^{(l)} = \sigma\Big(\sum\nolimits_{j \in \mathcal{N}(i)} \beta_{ij}^{(l)} \cdot \mathbf{tp}_j^{(l)}\Big) \tag{16}$$

where $\mathbf{a}_N$ and $\mathbf{W}_7^{(l)}$ are the topology specific attention vector and trainable weight matrix respectively, $\beta_{ij}^{(l)}$ defines the attention weight score which indicates the importance of neighbor node V_j and $\mathbf{ht}_i^{(l)}$ is the neighbourhood topology representation of target node V_i in the l-th layer.

3.3 Interaction module

In the above two modules, we consider the different contributions of attribute and topology of neighbourhood to the target node representation, and independently learn the neighbourhood attribute representation and neighbourhood topology representation of the target node. In this section, we explore the interaction and correlation between the two kinds of representations of the target node in current layer to obtain the global representation of the next layer. It can be formulated as follows:

$$\mathbf{h}_i^{(l+1)} = \mathbf{W}_8^{(l)} \cdot \mathbf{hf}_i^{(l)} + \mathbf{W}_9^{(l)} \cdot \mathbf{ht}_i^{(l)} + \mathbf{M}_T^{(l)} \cdot \mathbf{h}_i^{(l)} \tag{17}$$

where $\mathbf{W}_8^{(l)}, \mathbf{W}_9^{(l)}$ are two type-specific weight matrices, $\mathbf{M}_T^{(l)}$ is the layer-specific transformation matrix introduced in topology aggreagtion module and $\mathbf{h}_i^{(l+1)}$ is the $l+1$ layer global representation of node V_i. After several API-GNN iterations, the node global representation $\mathbf{H}^L \in \mathbb{R}^{N \times d}$ of the L-th layer is used as node final representation.

3.4 Optimization objective

The final prediction result $\hat{\mathbf{Y}}$ of API-GNN for various downstream tasks can be derived from node final representation $\mathbf{H}^L$ via a single layer nonlinear projection. Specifically, we select node classification and link prediction as downstream tasks to evaluate model performance in our paper, the model predictions of this two tasks $\hat{\mathbf{Y}}_{nc}$, $\hat{\mathbf{Y}}_{lp}$ can be formulated as follow:

$$\hat{\mathbf{Y}}_{nc} = softmax(\sigma(\mathbf{H}^L \mathbf{W}_{nc})) \tag{18}$$

$$\hat{\mathbf{Y}}_{lp} = softmax(\sigma((\mathbf{H}^L_{src}||\mathbf{H}^L_{dst})\mathbf{W}_{lp})) \tag{19}$$

where $\hat{\mathbf{Y}}_{nc}$ and $\hat{\mathbf{Y}}_{lp}$ are the model prediction results for node classification and link prediction respectively. $\mathbf{W}_{nc} \in \mathbb{R}^{d\times|\mathcal{C}|}$, $\mathbf{W}_{lp} \in \mathbb{R}^{2d\times 2}$ are the parameters of the single layer nonlinear projection. $\mathbf{H}^L_{src}$, $\mathbf{H}^L_{dst}$ are the node final representation matrices of edge endpoints.

After obtaining the model prediction, the supervised loss functions of API-GNN for node classification task and link prediction task can be defined as:

$$L_{nc} = -\frac{1}{|\mathcal{Y}_{\mathcal{C}}|}\sum_{i\in\mathcal{Y}_{\mathcal{C}}}\sum_{c}^{|\mathcal{C}|} y_{ic}\ln\hat{y}_{ic} \tag{20}$$

$$L_{lp} = \frac{1}{|\mathcal{Y}_+||\mathcal{Y}_-|}\sum_{i,j\in\mathcal{Y}_+\cup\mathcal{Y}_-}(y_{ij}\ln\hat{y}_{ij} + (1-y_{ij})\ln(1-\hat{y}_{ij})) \tag{21}$$

For node classification task, we randomly select some labeled nodes to construct the training set $\mathcal{Y}_{\mathcal{C}}$ and $\mathcal{C}$ is the set of node label. For link prediction task, we randomly select some real-edges and some non-existent edges to construct the postive instances link set $\mathcal{Y}_+$ and negative instances link set $\mathcal{Y}_-$.

Then the total loss function of node classification or link prediction task based on API-GNN is as follows:

$$L_{all} = \lambda_1 L_{\text{sup}} + \lambda_2 L_{auto} + \lambda_3 L_{reg} \tag{22}$$

It can be divided into three parts: L_{sup} is the loss of supervised downstream task such as L_{nc}, L_{lp} in this paper. L_{auto} is the auto-encoder loss in topology aggregation module and L_{reg} is the l_2-normalization regularization loss to prevent overfitting. $\lambda_1, \lambda_2, \lambda_3$ are the corresponding weight coefficients.

4 Experiments

4.1 Experiment setups

4.1.1 Datasets

In order to verify the effect of API-GNN, we conduct extensive experiments on six real-world attributed network datasets. Two citation networks ACM, DBLP [18], two social networks Facebook, Google+ [12], and two e-commerce networks Amazon [13], JD [2]. Their statistical information is summarized in Table 2. It is noticeable that we encode node features and use 0 as padding value to obtain the original attribute vectors with the same dimensional.

- *A*CM: In this dataset, we select papers as nodes and use metapath paper-author-paper to construct edges. Paper features correspond to elements of a bag-of-words represented of keywords.
- *D*BLP: Different from ACM, the node in DBLP network represents author and edge exists when the two authors publish their paper at the same conference. Author features are the elements of a bag-of-words represented of keywords.

Table 2 Statistics of datasets used in this paper

Datasets	Nodes	Features	Links	$\bar{d}$	Labels
ACM	3025	1870	6454	4.27	3
DBLP	3890	334	2347227	1206.80	4
Facebook	3401	1024	71254	41.90	2
Google+	5807	640	885285	304.90	2
Amazon	5004	354	27313	10.92	5
JD	5289	3496	551825	208.67	4

- *F*acebook: We regard following relationship as the edge and transform it into undirection relation. Notice that all of node attributes are anonymized. We select birthday, education, name, language, locale, hometown, location and work as node features.
- *G*oogle+: This dataset is similar to Facebook, but the attributes are not anonymized. For node features, we select place, name, institution, university and job.
- *A*mazon: We build the subgraph of clothing items in `Amazon.com` and use co-rating relation as edge. The rank, price, brand, title, description, dimension and weight are selected as item features.
- *J*D: For JD dataset, we also select clothing items as nodes and construct edges by co-click and co-order relation. The features of item include class-id, brand, price, description.

4.1.2 Baselines

We compare our method with three kinds of strong baselines: conventional graph embedding, conventional GNN-based graph embedding and improved GNN-based graph embedding. Futhermore, we also design three variants of API-GNN to demonstrate the ability of different modules of our method.

- *C*onventional Graph Embedding: we select three methods based on random walk: **Deepwalk** [14], **Node2vec** [4] and **GraphRNA** [7]. We also select three graph embedding methods based on exploiting proximity: **LINE** [17], **SDNE** [21], **DANE** [3]. **Struc2Vec** [15] which considers the spatial structure similarity is another baseline.
- *C*onventional GNN-based Graph Embedding: we use two average aggregation models: **GCN** [9] and **GraphSAGE** [5]. **GAT** [20] is also selected as baseline which introduces attention mechanism to neighborhood aggregation .
- *I*mproved GNN-based Graph Embedding: **CS-GNN** [6] designs a new mechanism to aggregate information from dissimilar neighbors. **GResNet** [27] explores variations of residual connections and **ALaGCN** [24] uses an adaptive layer to achieve a better aggregation effect. **DAGNN** [10] decoupling representation transformation and propagation in graph convolution operation.
- *V*ariants of API-GNN: we design three variants **A-GNN**, **T-GNN** and **AP-GNN** of API-GNN. Different from API-GNN, A-GNN only considers the original attribute aggregation module and ignores the topology aggregation module; while T-GNN contains both the original attribute aggregation module and the topology aggregation module, but its original attribute aggregation module only has node-level attention; the original attribute aggregation module and topology aggregation module are retained in AP-GNN, but

the iteraction module of each iteration is ignored, and the interaction is only carried out in the last iteration.

4.1.3 Smoothness metrics

In order to quantify the attribute disturbance in neighborhood aggregation of graph neural networks, we calculate Smoothness Metric Value (SMV) to for our model and other GNN-based baselines. SMV is proposed by [10] and it is a smoothness metric which reflects the similarity of node representations with Euclidean distance. The larger the distance (the larger the SMV value), the lower the smoothness (the lower the similarity).

It is defined as follow:

$$D(h_i, h_j) = \frac{1}{2} \| \frac{h_i}{\|h_i\|} - \frac{h_j}{\|h_j\|} \| \tag{23}$$

$$SMV_i = \frac{1}{n-1} \sum\nolimits_{j \in V, j \neq i} D(h_i, h_j) \tag{24}$$

$$SMV_{\mathcal{G}} = \frac{1}{n} \sum\nolimits_{i \in V} SMV_i \tag{25}$$

where h_i is the node global representation learned by graph neural network of node V_i and $\| \cdot \|$ is the Euclidean norm. We use $SMV_{\mathcal{G}}$ to measure the overall smoothness of an attributed graph $\mathcal{G}$, because $SMV_{\mathcal{G}}$ represents the distance between all the node representations of $\mathcal{G}$. Therefore, the larger the $SMV_{\mathcal{G}}$, the more dissimilar the node representations of $\mathcal{G}$, the smaller the overall smoothness of $\mathcal{G}$, i.e., the smaller the disturbance to the original attribute. On the contrary, the smaller the $SMV_{\mathcal{G}}$, the greater the overall smoothness of $\mathcal{G}$, and the greater the disturbance to the original attributes.

4.1.4 Parameters

We use the same set of hyper-parameters for API-GNN and GAT. The node representation dimension is set to 32 and the weight coefficients $\lambda_1, \lambda_2, \lambda_3$ are set to 0.5, 0.3, 0.2 respectively. We set the learning rate to 0.001 and select Adam as optimizer. For other models, we use their reported default parameters. All models are trained for 2000 epochs with an early stopping strategy based on both convergence behavior and accuracy of the validation set.

4.2 Node classification

For the node classification task, we select research area, gender and price as the node labels of citation network, social network and e-commerce network respectively. In all datasets, 80% nodes are used as the training set, 10% nodes are used as the validation set and the rest as the test set. The accuracies on test sets are shown in Table 3.

Compared with the conventional graph embedding, conventional GNN-based graph embedding and improved GNN-based graph embedding methods, our method API-GNN improves the accuracy by 2.8%, 3.3% and 1.5% respectively on six datasets. This shows that API-GNN can learn better node representation by solving the problems of attribute

Table 3 Accuracy on the six datasets for node classification

Methods	ACM	DBLP	Facebook	Google+	Amazon	JD
DeepWalk	0.713	0.920	0.633	0.826	0.457	0.416
Node2vec	0.723	0.864	0.630	0.830	0.465	0.431
GraphRNA	0.934	0.884	0.654	0.826	0.475	0.493
LINE	0.908	0.918	0.648	0.835	0.449	0.410
SDNE	0.901	0.918	0.628	0.821	0.459	0.403
Struc2vec	0.686	0.812	0.584	0.802	0.449	0.374
DANE	0.799	0.794	0.663	0.804	0.477	0.342
GCN	0.888	0.900	0.666	0.804	0.477	0.363
GraphSAGE	0.875	0.905	0.657	0.790	0.461	0.482
GAT	0.911	0.920	0.689	0.804	0.477	0.461
GresNet	0.861	0.910	0.657	0.790	0.479	0.533
CS-GNN	0.901	0.918	0.669	0.793	0.479	0.509
ALaGCN	0.927	0.923	0.704	0.814	0.487	0.535
DAGNN	0.927	0.906	0.666	0.828	0.487	0.488
A-GNN	0.917	0.925	0.692	0.818	0.487	0.505
T-GNN	0.931	0.928	0.710	0.833	0.491	0.522
AP-GNN	0.924	0.920	0.704	0.826	0.477	0.520
API-GNN	**0.944**	**0.933**	**0.716**	**0.845**	**0.495**	**0.548**

disturbance and the influence differences of attribute and topology on node representation in GNN aggregation. In order to further verify the performance of API-GNN, we compare the performances of three variants of API-GNN (A-GNN, T-GNN and AP-GNN) with improved GNN-based methods. A-GNN is superior to the improved GNN-based methods in most datasets, which shows that it is very important to consider both attribute-level attention and node-level attention for attributed networks. T-GNN is better than A-GNN, which indicates that it is better to consider the influences of topology and attribute on node representation than to consider attribute only. However, the performance of AP-GNN is not better than that of A-GNN and T-GNN, which indicates that the interaction of attribute and topology information in each interation is also very important for node representation.

4.3 Link prediction

For link prediction task, we randomly hide 10% edges for validating, 10% edges for testing and train node representation on the rest network. We regard the real-edges as positive instances and randomly select 10 non-existent edges as negative instances for each positive instance. We use the concatenating of node pair representations as input and construct the Multilayer Perception (MLP) to predict the edges of test set. Area Under Curve (AUC) is adopted to quantify the performance and the experiment results are shown in Table 4.

It is the same as the performance on the node classification task, API-GNN performs consistently much better than all baselines on all datasets. On average, compared with the conventional graph embedding and conventional GNN-based graph embedding and improved GNN-based graph embedding methods, our method API-GNN improves the accuracy by 9.5%, 7.0% and 1.8% respectively on six datasets. It validates the effectiveness of our proposed model architecture once more. Moreover, we compare A-GNN

Table 4 AUC on the six datasets for link prediction

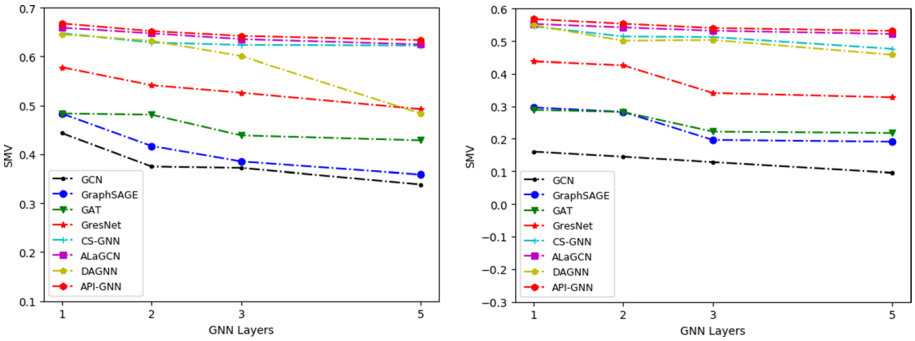
Methods	ACM	DBLP	Facebook	Google+	Amazon	JD
DeepWalk	0.555	0.537	0.574	0.584	0.586	0.559
Node2vec	0.552	0.543	0.572	0.577	0.590	0.555
GraphRNA	0.687	0.547	0.693	0.764	0.774	0.746
LINE	0.559	0.542	0.573	0.566	0.596	0.553
SDNE	0.531	0.525	0.546	0.557	0.570	0.537
Struc2vec	0.547	0.543	0.579	0.593	0.607	0.566
DANE	0.640	0.606	0.651	0.651	0.606	0.635
GCN	0.734	0.574	0.724	0.760	0.756	0.736
GraphSAGE	0.726	0.564	0.713	0.813	0.730	0.706
GAT	0.697	0.536	0.671	0.820	0.778	0.694
GresNet	0.658	0.641	0.746	0.832	0.802	0.774
CS-GNN	0.706	0.675	0.777	0.882	0.787	0.777
ALaGCN	0.737	0.666	0.755	0.885	0.814	0.788
DAGNN	0.744	0.689	0.775	0.894	0.822	0.797
A-GNN	0.712	0.645	0.733	0.838	0.796	0.754
T-GNN	0.739	0.686	0.762	0.883	0.816	0.788
AP-GNN	0.726	0.636	0.751	0.868	0.787	0.766
API-GNN	**0.756**	**0.707**	**0.786**	**0.894**	**0.836**	**0.804**

with conventional GNN-based graph embedding methods and find that the former outperform the latter. It indicates that modeling the different node attributes influence in neighborhood aggregation improves link prediction ability. T-GNN achieve the similar performance comparing with improved GNN-based methods, which indicates that the importance of topology aggregation in link prediction task. The performance of API-GNN decreases severely without interaction module, it proves once again the important role and irreplaceable of interaction module of API-GNN.

4.4 Attribute disturbance analysis

As mentioned in Section 5.1, because the smoothness of node representations is caused by attribute disturbance, the smaller the $SMV_{\mathcal{G}}$, the bigger the attribute disturbance in neighborhood aggregation.

As shown in Figure 4, our API-GNN is superior to all baselines in the aspect of eliminating attribute disturbance. Conventional graph neural networks such as GCN, GraphSAGE and GAT don't consider the attribute disturbance problem in neighborhood aggregation, so the $SMV_{\mathcal{G}}$ of these methods is very small, which means that the node representation obtained by these methods is easy to oversmooth. Improved graph neural networks such as GresNet and DAGNN alleviate the attribute disturbance problem by designing a sophisticated neighborhood aggregation. Therefore, their $SMV_{\mathcal{G}}$s are larger than that of the conventional GNN method. Our method is superior to the improved GNN, which proves that our proposed strategy, i.e. original attribute aggregation strategy, is better than the sophisticated neighborhood aggregation methods.



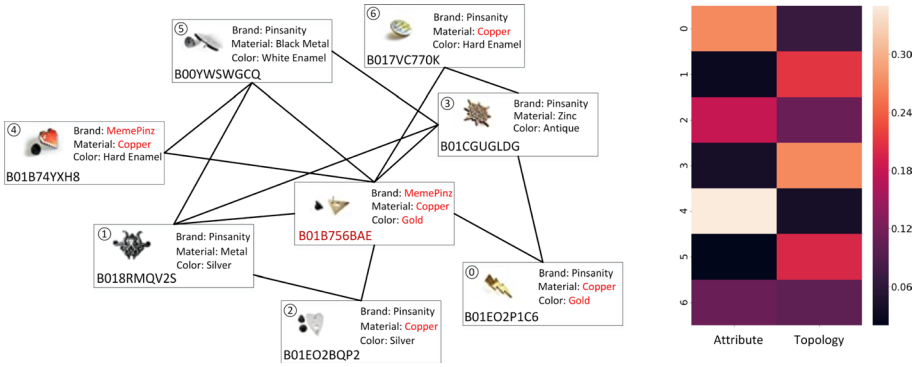
(a) ACM (b) Facebook

Fig. 4 Attribute Disturbance analysis of different methods

It can also be seen from Figure 4 that with the increase of GNN iteration layers, the decrease of $SMV_{\mathcal{G}}$ is very small, which can be almost ignored. This proves the robustness of the proposed strategy in avoiding attribute disturbance.

4.5 Impact difference analysis

In order to intuitively show the influence of neighbor topology and attributes on the global representation of the target node in API-GNN, we randomly sample an item B01B756BAE as the target node and extract its ego-network (Figure 5(a)) from Amazon dataset. In Figure 5(b), we use the hot-map to show the attention distribution (obtained from API-GNN) of different neighbors' attribute and topology on the target node representation. The lighter the color in the hot-map, the higher the attention of the neighbor node on the attribute or topology. In Figure 5(b), the colors of node 0 (B01EO2P1C6) and node 4 (B01B74YXH8) in the attribute block of hot-map are very light, which indicates that the attribute information dominates the information that these two nodes propagate to the target node in the aggregation process. As we can see that in Figure 5(a), node 0 and node 4 have



(a) (b)

Fig. 5 Influence difference of neighbor attribute and topology on global representation of target node

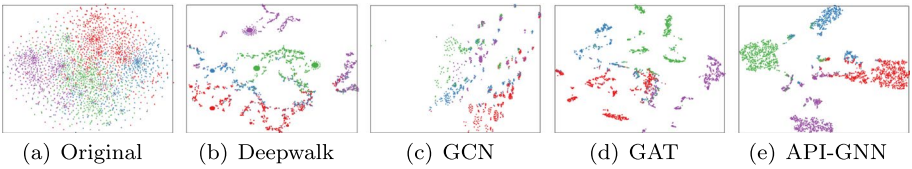


Fig. 6 2D visualization of node representation on DBLP using t-SNE. The different colors represent different classes

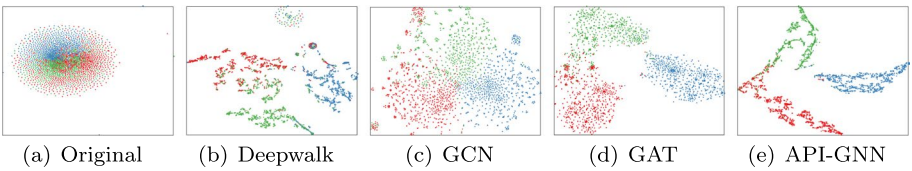


Fig. 7 2D visualization of node representation on ACM using t-SNE. The different colors represent different classes

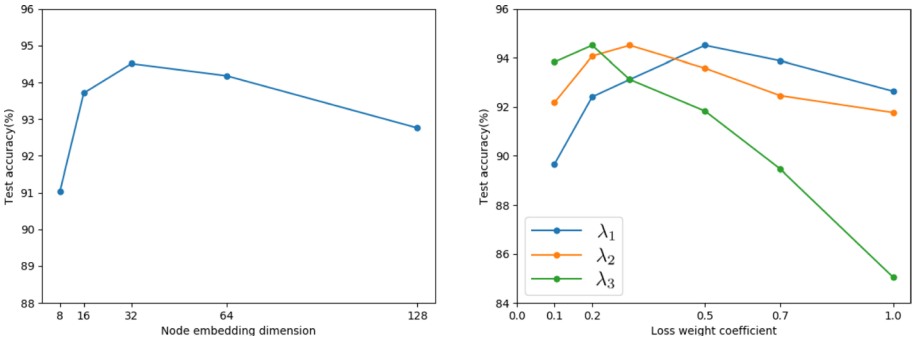
more similar attributes to the target node. Conversely, in Figure 5(b), the colors of node 1 (B018RMQV2S) and node 5 (B00YWSWGCQ) in the topology block of hot-map are very light, which indicates that the topology information of two nodes is more important to the target node in the aggregation process. As we can see that in Figure 5(a), there are more neighbor nodes to share among those nodes.

4.6 Visualization

For a more intuitive comparison and to further show the effectiveness of our proposed model, we conduct the task of visualization on ACM and DBLP dataset. Specifically, we learn the node representations based on API-GNN and other baselines, then we utilize t-SNE [11] to project the learned node representations into a 2-dimensional space. The results in Figures 6 and 7 are colored by real labels. From the visualization, we can see that API-GNN performs best as the learned node representations have the hightest intra-class similarity and the clearest distinct boundaries among different classes.

4.7 Parameter sensitivity analysis

In this section, we investigate the parameter sensitivity (Figure 8). More specifically, we evaluate how different numbers of the node representation dimension d and different values of loss weight coefficients $\lambda_1, \lambda_2, \lambda_3$ can affect the results. We report accuracy of node classification on the dataset of ACM, similar results are observed on other datasets and other tasks as we omit them. From the results in Figure 8, we can find that the performance of API-GNN increases as d increases from 8 to 32 and API-GNN achieve the best performance as $d = 32$. When d continues to increase, the performance of API-GNN will decrease, because too large d will lead to over fitting problem. For loss weight coefficients, the optimal settings of $\lambda_1, \lambda_2, \lambda_3$ are 0.5,0.3,0.2 respectively. It demonstrates that



(a) Dimension d (b) Coefficient $\lambda_1, \lambda_2, \lambda_3$

Fig. 8 The impacts of parameters the node representation dimension d and loss weight coefficients $\lambda_1, \lambda_2, \lambda_3$

the supervised loss is the most important loss and the regularized loss has the least effect on the model.

5 Related work

Our model aims at learning node representation on attributed network. At present, the mainly graph embedding methods are conventional graph embedding and GNN-based graph embedding. For convolutional GNN-based graph embedding, there are some problems ignored such as over-smoothing and some researches have been proposed to address those problem. Therefore, we can divide GNN-based graph embedding into two categories: conventional GNN-based graph embedding and improved GNN-based graph embedding.

5.1 Conventional graph embedding

Early researches on graph embedding merely focus on finding node similarity with graph structure. Perozzi et al. propose DeepWalk [14] which deploys truncated random walks on nodes to learn node representation. Node2vec [4] extends DeepWalk with more sophisticated random walk and breadth-first research schema. Different from the random walk model, LINE [17] separately exploits the first-order proximity and second-order proximity to capture both the local and global network structure. SDNE [21] follows and extends LINE, it designs an AutoEncoder and jointly optimizes first-order proximity and second-order proximity to learn graph embedding. Struc2Vec [15] consider that there is a high similarity between nodes that are not neighbor nodes. It adopts spatial structure similarity to learn graph node representation. Most of conventional graph embedding methods only use graph topology information and ignore node attribute information, However, node attribute information is important auxiliary information for graph embedding task and ignoring it will damage model performance. There are also some works use both node attribute information and graph structure information such as GraphRNA [7] and DANE [3] and so on. GraphRNA explores the random walk in attribute graph and topology graph, DANE preserves various proximities in topological structure and node attributes. However, those methods can't make full use of the attribute information of neighbor nodes.

5.2 Conventional GNN-based graph embedding

Recently, more attention are paid to applying Graph Neural Networks to learn node representation due to their amazing performances. Bruna et al. [1] first designs the graph convolution operation in Fourier domain by the graph Laplacian. Kipf and Welling [9] simplify previous methods via first-order approximation of localized spectral filters on graphs to reduce the complexity of convolution. In order to improve the generalization ability of GNNs, GraphSAGE [5] proposes to sample and aggregate attributes from a node's local neighborhood with mean/max/LSTM. GAT [20] considers the influence of different neighbor nodes in neighborhood aggregation and introduces the attention mechanism to aggregate node attributes with the learned weights. Although these conventional GNNs can use both attribute information and topology structure information, they suffer from the deviation of original attribute and the decrease of node discriminability during neighborhood aggregation.

5.3 Improved GNN-based graph embedding

In order to address the problems of conventional GNN-based methods, some researches have been proposed to improve the neighborhood aggregation in conventional GNNs. CS-GNN [6] focuses on aggregating information from dissimilar neighbors and Xie et al. [24] designs a intricate neighborhood aggregation which drops some noise nodes to avoid that all nodes learn the identical node representation. Meanwhile, GResNet [27] explores variations of residual connections and DAGNN [10] decouple the representation transformation and propagation in graph convolution operations for increasing the depth of GNN model. However, none of these models explicitly takes into account the attribute disturbance and different effects of attribute and topology on node representation.

6 Conclusion

In this paper, we analyse ubiquitous problems in most existing GNN-based models: the original attribute deviation and the influence differences of attribute and topology on node representation. To overcome this vital issue, we proposed a novel API-GNN model. Extensive experiments on six real-world datasets demonstrate the rationality and effectiveness of API-GNN.

In the future work, an potential direction is to extend the API-GNN to heterogeneous network which contains different types of nodes and edges.

Acknowledgements This work was supported by the the Youth Innovation Promotion Association CAS (No.2018192) and the National Natural Science Foundation of China (No. 61902394).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and deep locally connected networks on graphs. In: International Conference on Learning Representations (2014)
2. Chen, W., Gu, Y., Ren, Z., He, X., Xie, H., Guo, T., Yin, D., Zhang, Y.: Semi-supervised user profiling with heterogeneous graph attention networks. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 2116–2122 (2019)
3. Gao, H., Huang, H.: Deep attributed network embedding. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 3364–3370 (2018)
4. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
5. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Annual Conference on Neural Information Processing Systems, pp. 1024–1034 (2017)
6. Hou, Y., Zhang, J., Cheng, J., Ma, K., Ma, R.T.B., Chen, H., Yang, M.: Measuring and improving the use of graph information in graph neural networks. In: International Conference on Learning Representations (2020)
7. Huang, X., Song, Q., Li, Y., Hu, X.: Graph recurrent networks with attributed random walks. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 732–740 (2019)
8. Jin, J., Qin, J., Fang, Y., Du, K., Zhang, W., Yu, Y., Zhang, Z., Smola, A.J.: An efficient neighborhood-based interaction model for recommendation on heterogeneous graph. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 75–84 (2020)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017)
10. Liu, M., Gao, H., Ji, S.: Towards deeper graph neural networks. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 338–348 (2020)
11. Visualizing data using t-sne: Maaten, L.v.d., Hinton, G. Journal of Machine Learning Research **9**(Nov), 2579–2605 (2008)
12. McAuley, J.J., Leskovec, J.: Learning to discover social circles in ego networks. In: Annual Conference on Neural Information Processing Systems, pp. 548–556 (2012)
13. McAuley, J.J., Targett, C., Shi, Q., van den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the International Conference on Research and Development in Information Retrieval, pp. 43–52 (2015)
14. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
15. Ribeiro, L.F.R., Saverese, P.H.P., Figueiredo, D.R.: struc2vec: Learning node representations from structural identity. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 385–394 (2017)
16. Sun, J., Guo, W., Zhang, D., Zhang, Y., Regol, F., Hu, Y., Guo, H., Tang, R., Yuan, H., He, X., Coates, M.: A framework for recommending accurate and diverse items using bayesian graph convolutional neural networks. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 2030–2039 (2020)
17. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: Proceedings of the International Conference on World Wide Web, pp. 1067–1077 (2015)
18. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 990–998 (2008)

19. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P.P.: Composition-based multi-relational graph convolutional networks. In: International Conference on Learning Representations (2020)
20. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
21. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 1225–1234 (2016)
22. Wu, J., He, J., Xu, J.: Demo-net: Degree-specific graph neural networks for node and graph classification. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 406–415 (2019)
23. Wu, Y., Lian, D., Jin, S., Chen, E.: Graph convolutional networks on user mobility heterogeneous graphs for social relationship inference. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 3898–3904 (2019)
24. Xie, Y., Li, S., Yang, C., Wong, R.C., Han, J.: When do gnns work: Understanding and improving neighborhood aggregation. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1303–1309 (2020)
25. Xu, C., Cui, Z., Hong, X., Zhang, T., Yang, J., Liu, W.: Graph inference learning for semi-supervised classification. In: International Conference on Learning Representations (2020)
26. Yang, L., Chen, Z., Gu, J., Guo, Y.: Dual self-paced graph convolutional network: towards reducing attribute distortions induced by topology. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 4062–4069 (2019)
27. Zhang, J., Meng, L.: Gresnet: Graph residual network for reviving deep gnns from suspended animation. arXiv:1909.05729 (2019)
28. Zhuang, C., Ma, Q.: Dual graph convolutional networks for graph-based semi-supervised classification. In: Proceedings of the International Conference on World Wide Web, pp. 499–508 (2018)

Authors and Affiliations

Yuchen Zhou[1,2] · Yanmin Shang[1,2] · Yanan Cao[1,2] · Qian Li[3] · Chuan Zhou[1,4] · Guandong Xu[3]

Yuchen Zhou
zhouyuchen@iie.ac.cn

Yanan Cao
caoyanan@iie.ac.cn

Qian Li
Qian.Li@uts.edu.au

Chuan Zhou
zhouchuan@amss.ac.cn

Guandong Xu
Guandong.Xu@uts.edu.au

[1] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

[3] University of Technology Sydney, Sydney, Australia

[4] Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China