SGNN: A Graph Neural Network Based Federated Learning Approach by Hiding Structure

Guangxu Mei
Shandong University
Jinan, China

Ziyu Guo
Shandong University
Jinan, China

Shijun Liu*
Shandong University
Jinan, China
lsj@sdu.edu.cn

Li Pan*
Shandong University
Jinan, China
panli@sdu.edu.cn

***Abstract*—Networks are general tools for modeling numerous information with features and complex relations. Network Embedding aims to learn low-dimension representations for vertexes in the network with rich information including content information and structural information. In recent years, many models based on neural network have been proposed to map the network representations into embedding space whose dimension is much lower than that in original space. However, most of existing methods have the following limitations: 1) they are based on content of nodes in network, failing to measure the structure similarity of nodes; 2) they cannot do well in protecting the privacy of users including the original content information and the structural information. In this paper, we propose a similarity-based graph neural network model, SGNN, which captures the structure information of nodes precisely in node classification tasks. It also takes advantage of the thought of federated learning to hide the original information from different data sources to protect users' privacy. We use deep graph neural network with convolutional layers and dense layers to classify the nodes based on their structures and features. The node classification experiment results on public data sets including Aminer coauthor network, Brazil and Europe flight networks indicate that our proposed model outperforms state-of-the-art models with a higher accuracy.**

***Index Terms*—network embedding, deep graph neural networks, node classification, federated learning**

I. Introduction

Graph is the primary representation of data with node attributes and local topological structures. For instance, graph data can be used to analyse the latent relations in social networks [1]–[3]. In general, the graph data contains structural information and content information with a high dimension. To make representation and computation easier, lots of models have been proposed for reducing the dimension of the graph data. Adjacency matrix is often used to measure the structure of a graph. In an unweighted graph, element in adjacency matrix is valued 1 if there is an edge between two nodes, valued 0, otherwise. In a weighted graph, element in adjacency matrix is the weight value between two nodes. However, the direct connection cannot reveal the structural relation among nodes precisely. Also, the adjacency matrix fails to describe the relation of two nodes without an edge. Thus, many methods have been proposed to describe the structural similarity instead of original adjacency matrix. LINE [4] uses node's first-order proximity and second-order proximity simultaneously to distinguish nodes that are different at structure. The first-order proximity is defined as the proximity of two nodes with a direct link. The second-order proximity describes the proximity of nodes sharing some common neighbors. While, LINE fails to leverage high-order proximity of nodes. GraRep [5] defines the k-order proximity to measure the global structure to make up for the consideration of high-order proximity. Struc2vec [6] constructs a multi-layer graph using hierarchy to measure the similarity of pairs of nodes, considering the structure as the unique identification of a node.

978-1-7281-0858-2/19/$31.00

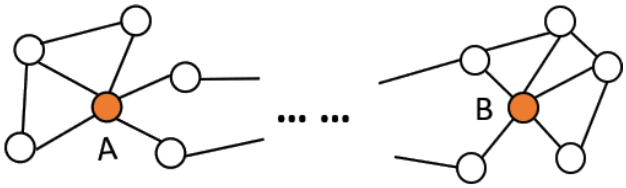

Fig. 1. An example of two similar nodes far away from each other.

Recently, by the great success of deep learning, graph neural networks like GCN [7], have become main methods to deal with node and graph classification tasks. The key of GCN lies in that it can learn low-dimension representations of nodes by aggregating information from nodes' local neighbors. GATs [8], Graph attention networks, use self-attention machine to attend a node's neighborhoods' features. However, models like GCN and its derivative models fail to measure the similarity of nodes far away from each other in the graph, which may be similar in structure. For instance, As shown in Fig. 1, the degree list of neighbors of node A should be $(2, 3, 2, 2, 2)$, which are the same as it of B. Thus, node A and B should have the similar representations though they are far from each other, because they have the same high-distance structural information. While, GCN fails to capture the structural information mentioned above. So, it is necessary to learn node representation by capturing both content information and structural information.

The motivation of this paper is that the model should be proposed to capture the structure of nodes far away from others precisely and then complete tasks. Moreover, users' privacy should be protected well during the process of collecting data and training models. For instance, data

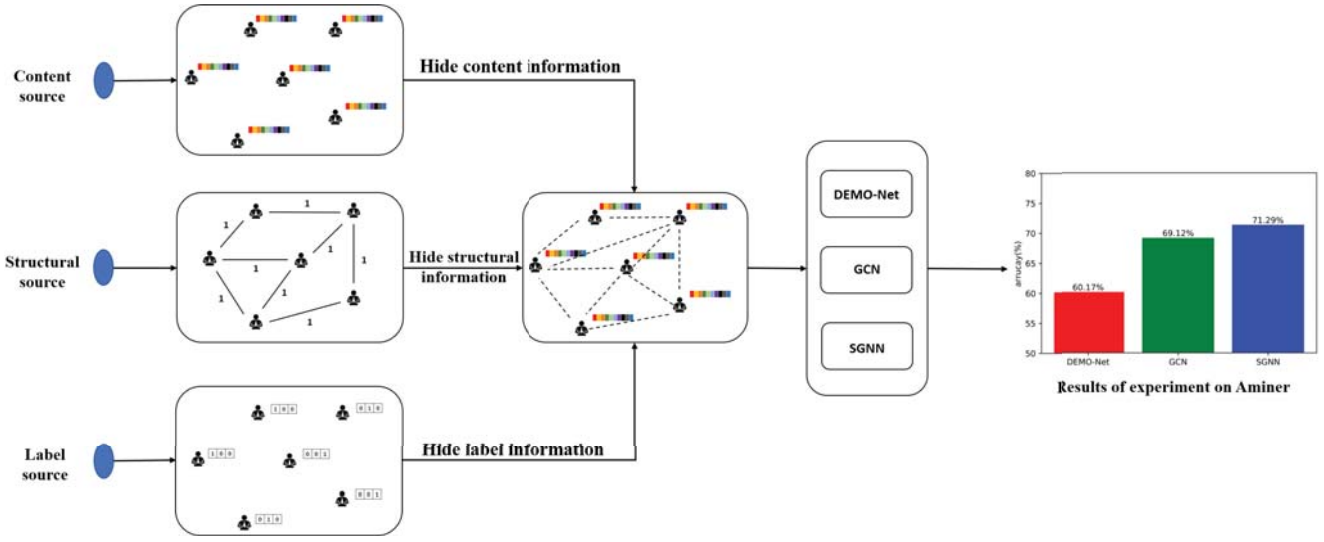

Fig. 2. The figure is an example of classification tasks taking the advantage of federated learning on Aminer data set. The left shows three data sources provide original content information, structural information and label information respectively. The content information denotes the users' private features. The structural information denotes the co-author relation among authors. The label information denotes users' labels. After hiding these information, the new graph is shown in the middle. The right part shows the results of the classification tasks. As is revealed, SGNN outperforms the other two models at least 2%.

from users' mobile phones is collected and sent to models to complete various tasks, and then these data are modified because parameters are trained by models. During this process, users' information are delivered without any encryption and modified because of the updating of parameters. The result is that users' information may be leaked. To prevent the loss of information, federated learning is proposed to protect users' privacy. Federated learning [9] is a kind of machine learning methods whose goal is to learn a distributed model to solve various tasks. Meanwhile, federated learning guarantees the information security and personal data privacy. Federated learning framework can be divided into data from different sources and the collaborator. Data from different sources cannot exchange information among them, but should be encrypted to protect users' privacy. The collaborator collects hidden information from those sources and trains models to complete the tasks. In our work, we take advantage of the way that federated learning can hide the original information, so that users' privacy can be protected.

In this paper, we propose a similarity-based graph neural network, SGNN, to redefine the distance among nodes and compute the similarity of nodes. Then, the similarity matrix is used to take place of adjacency matrix and then it is input to the graph neural networks. Meanwhile, as shown in Fig. 2, original content information, structural information and label information are from three different data sources, each of which need not to exchange messages with any other data source. The content information denotes the users' private features. The structural information denotes the relations like collaborative relations among authors in academic networks. The label information denotes users' labels. The original information are encrypted and then they are sent to the collaborator models like DEMO-Net [10], GCN [7] and SGNN (our work). The collaborators deliver training parameters to the hidden information and receive the messages from them. After learning the model, the results are given. In this paper, SGNN belongs to the vertical federated learning, in which data from different sources have most public users and have the same training task. To hide the private information, we first pick out users that all data sources have. Then we hide the structural information by computing node similarity matrix to take the place of the adjacency matrix, and hide the content information by constructing one-hot encoding matrix instead of the original feature matrix. We input the similarity matrix and one-hot encoding matrix into our proposed model, SGNN, to improve the classification accuracy, which are higher than the accuracy of GCN.

In summary, this paper makes several contributions, which are shown as follows:

- It proposes a model, SGNN, which constructs a similarity matrix to capture the high-distance structure precisely and overcomes the limitation that models can only capture local structure in the previous models like Struc2vec [6], DEMO-Net [10] and GCN [7].
- It takes advantage of the thought of federated learning, especially vertical federated learning, to hide the private information of users. By using the similarity matrix, it hides the structural information. Moreover, it hides the content information using one-hot encoding method.
- It conducts the node classification experiments on different data sets including large-scale graph and small-scale graph. The results of experiments reveal superior performance of SGNN. The accuracy of node classification is higher than existing models, especially on large-scale graph data sets.

II. Related Work

In recent years, network embedding has received more and more attention of researchers. It aims to learn a low dimension representation of nodes with high dimension features and complex structure. The technique can be traced back to the machine learning applications of NLP [11], which can be used in the field of network embedding. Most of previous works first compute a node representation by reducing the dimension of data, and then input the representation into NLP models. Word2vec [12] takes as its input a large corpus of text and produces a vector space with each unique word in the corpus being assigned a corresponding vector in the space. DeepWalk [13] uses word2vec to learn graph representation by constructing random walks sequences. It obtains the structural similarity on the basic of depth first search. The graph is regarded as the corpus of text and nodes in the graph are regarded as words in the corpus. Because the first-order proximity can only capture structure from 1-step neighbors instead of global structure, LINE [4] defines second-order proximity by computing the similarity of shared neighborhood structures of nodes. Each node in the graph is treated as two roles: the node itself and an context node of other nodes. LINE trains the model combining first-order and second-order proximity. While, LINE fails to take high-order proximity into consideration. To solve this problem, GraRep [5] defines probability transition matrix $A = D^{-1}S$, where D is the degree matrix of the graph and S is the adjacency matrix. The k-step probability transition matrix is defined as $P = A^k$, in which $p_k(i|j) = A^k_{i,j}$ denotes the probability for transition between v_i and v_j in k-steps. The methods above considered that two nodes with closer distance are more similar. In some real-world networks, however, nodes far from each other may also be similar because they have similar structure. Struc2vec [6] only measures the structural similarity instead of the direct connection of the original network. It constructs a hierarchical graph to leverage the structure and uses random walks to get node sequences. Then, a language model is learned to generate the representation of the graph. The advantage of Struc2vec is that the function of defining the structural similarity is effective, while the disadvantage is that the model neglects the content information of nodes. Metapath2vec [14] formalizes random walks in a heterogeneous network based on meta-path to construct neighborhoods of nodes, and then inputs them into Skip-gram [12] to perform network embeddings.

Graph neural networks have made great progress in node and graph classification tasks recently. Graph Convolutional Networks [7] learn node representation by aggregating messages from 1-step neighbors using mean pooling. GCN has the following limitations: 1) it cannot be efficiently used on large and dense graphs because of the memory demand; 2) it would be a time-consuming process that aggregating information from a node's neighbors especially when the number of neighbors is large. To limit the size of neighbors and reduce the computing complexity, GraphSAGE [1] samples certain number of nodes from 1-step neighbors and aggregates features using optional methods like mean aggregator, max aggregator and LSTM [15] aggregator. It does not study the node representation directly, while it studies the aggregation functions. For new nodes, the model generates the embedding representations directly instead of learning the model once more. Graph Attention Networks [8] computes the relation of nodes dynamically using self-attention mechanism. While, the models mentioned above neglect the nodes structural information. DEMO-Net [10] classifies nodes using the structural feature which has three properties: seed-oriented, degree-aware and order-free. The main thought of the model is that nodes have same degree should be similar. P-GNN [16] samples multiple anchor-sets with different sizes, and then computes the embeddings through getting messages from a given node and the anchor-sets. LDS [17] learns a discrete distribution on the edges of the network to complete the classification tasks in the condition that the part of edges are missing.

In this paper, we proposed a similarity-based graph neural network that captures the degree message of neighbors. Compared with the existing models, we pay more attention to the structure to make up for the loss of graph structure, and preserve the information in the feature space at the same time.

Recently, federated learning [9] [18] and related models are proposed as a general model that trains data from different devices and protect data privacy on each data source. For example, an instance of federated learning trains a classification/prediction model on a collaborator while keeping the training data on users' device like mobile phones. According to the different data sets, federated learning contains three kinds: horizontal federated learning, vertical federated learning and federated transfer learning. To adjust to data sets whose number of public uses is few, horizontal federated learning trains models through extracting features from completely different users. While, vertical federated learning deals with problems with data sets which have a large number of public users and few public features. It trains models using features from the same users. For data sets with few public users and features, federated transfer learning is used to get over the lack of public users and public features. Federated Transfer Learning [19] (FTL), learns a supervised learning model which can improve performance of every member model by transferring data in the network. Another contribution of FTL is that it takes data security into consideration in the process of training, evaluation and cross-validation. SecureBoost [20] is a novel tree-boosting system, with a high accuracy the same as methods without privacy protection. It exchanges intermediate computation among sub-models in the framework. Also, it guarantees no information of data providers is leaked. SecureBoost proves the leakage of information and security theoretically.

Methods mentioned above all have their own advantage. However, it is difficult to combine the advantages of them to solve our problem. In our work, taking advantage of the thought of federated learning, we capture the node similarity using graph neural networks to classify nodes in networks.

III. MODEL: SGNN

In this section, we introduce our proposed model, SGNN, to learn the representation of each node that captures structural information from its local neighbourhood and hide the content information and structural information at the same time. Therefore, SGNN should be designed with two following hypotheses:

- Users' private information should be protected carefully and users' content information and structural information could be represented separately. Instead of including its own features merely, the embedding of a node could be aggregated from information of its local neighborhood including their structure and features.To avoid leaking these private information, we use federated learning framework to solve our problem more suitably. In this way, original information is encrypted in our work.
- The similarity between two nodes could be negatively correlated to their distance in structure. In other words, the nodes with similar local structures could have similar representations in embedding space, while nodes should be far away from others because their local structure are much different. Therefore, we hide the structure information by computing the similarity among nodes instead of original structure.

To this end, we propose SGNN, a Similarity-based Graph Neural Network, which learns latent representations of nodes from neighborhoods' features and structural information. Meanwhile, SGNN prevents the leakage of users' information by the thought of federated learning. First, SGNN processes the content information by one-hot encoding. Then, SGNN computes the similarity among nodes. After that, the content information and structural information are input to complete the classification tasks.

To describe structural information more precisely, we define a new method to compute the similarity matrix in the pair of nodes. The computing methods of similarity matrix is shown in 1) and computing the aggregate feature vectors (content vectors) is shown in 2). After that, we train the graph neural networks to complete node classification tasks. The process of the model is shown in Fig. 3. SGNN contains the following steps:

1) Compute the structural distances of nodes. First, for each node in the graph, we get all its neighbors. Then, the degree of each neighbor is computed. After that, each node gets a degree list of its neighbors. We use DTW [21] method to compute the distance of these degree lists. The distance of nodes are measured as the similarity of their ordered degree lists. Compute the similarity matrix. The distances aforementioned are used to compute the similarity matrix of nodes. The pair of nodes with larger distance have smaller weight. The detail of computing is shown in the following section.
2) To hide the content information, one-hot encoding is used to map the original features to a matrix, each row of which denotes a user's features. Element in each row is 1 or 0.
3) Learn node representation through degree-based graph neural network. We improve the GCN to get a better result in node classification through adding dense layers and training the network with the weights matrix involving the degree message.

In the next following parts, we introduce the detailed process by separating the model into four parts.

A. Computing Similarity Matrix

According to the aforementioned hypotheses, we should hide the original structure of nodes. We redefine the distance of a pair of nodes. If two nodes' neighborhoods have similar structure, the distance between the two nodes should be closer than that of nodes with different neighborhoods' structure.

We define $G = (V, E)$ as the unweighted, undirected graph with the node set V and the edge set E. The number of nodes is denoted as $n = |V|$. The degree of node u is computed as shown in Eq. 1:

$$degree(u) = \Sigma_j A_{ij} \tag{1}$$

where A is the adjacency matrix and i is the index of node u in the graph.

For node u, we describe its structure as the degree lists of its neighbors. Let $N(u)$ denotes the list of node u's neighbors in which element is in range $[0, n-1]$.

Considering the sizes of neighbors of different nodes may be different, we use ordered degree list to present a node's structure which is convenient for computing the similarity of sequences with different size.

Let $s(N(u))$ denote the ordered degree list of node u's neighbors. So we compute the similarity of two nodes using the following function:

$$Distance(u, v) = f(s(N(u)), s(N(v)) \tag{2}$$

where f is the approach of measure the difference of $s(N(u))$ and $s(N(v))$.

The next step is to define f to compute the similarity of two nodes. Note that the length of ordered degree lists may be different.

Dynamic Time Warping (DTW) is one of the algorithms for measuring similarity between two temporal sequences, whose sizes may be different [21]. DTW has been used in the field like video and audio applications in which data can be turned into a linear sequence. For two sequences with size n and m, DTW first constructs a matrix $W \in \mathbb{R}^{n \times m}$, whose element is the euclidean distance of two nodes from the two sequences. Then, DTW finds a path across the matrix with minimized cumulative distance, using dynamic programming method. The cumulative distance is the distance of the two sequences. Thus, the distance of two sequences with different sizes can be computed. In this paper, we use DTW to measure the distance between $s(N(u))$ and $s(N(v))$. Specifically, the distance between two degree lists are defined as follows:

$$f(s(N(u)), s(N(v))) = DTW(s(N(u)), s(N(v))) \tag{3}$$

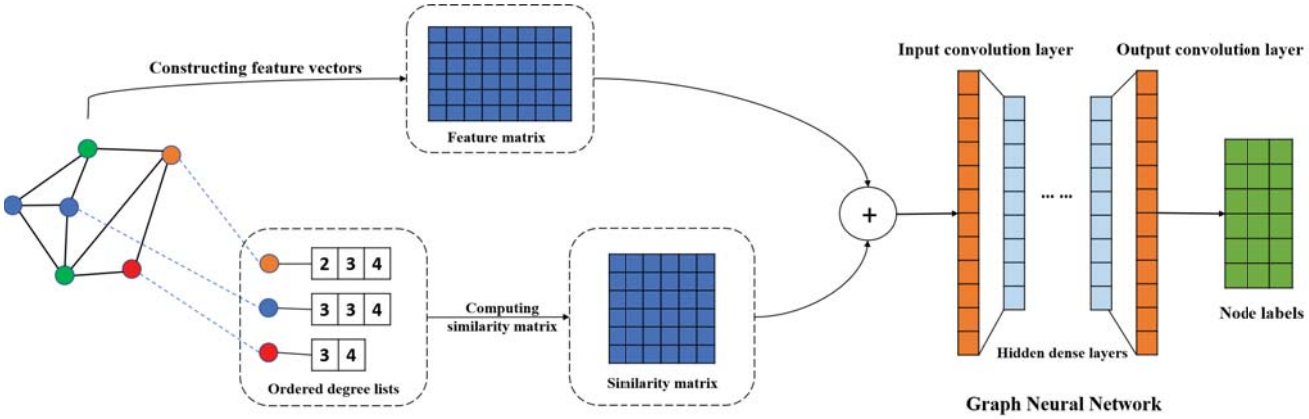

Fig. 3. Illustration of SGNN model. The model first gets the ordered degree list for each node in the graph. Then, using DTW method, SGNN computes similarity matrix, each element of which denotes the similarity of two degree lists. The similarity matrix is regarded as the nodes' similarity matrix. Then, the feature matrix and the similarity matrix are input into the graph neural network to classify nodes in the graph. The output is the predicted labels of nodes.

Therefore, combining Eq. 2 and 3, we obtain the distance of a pair of nodes, u and v.

Note that nodes with large distances should have low similarity. To map the similarity of two nodes into range[0, 1], we compute the similarity with Eq. 4:

$$Similarity(u, v) = e^{-Distance(u,v)} \tag{4}$$

where u and v are two nodes in original graph, and $Distance(u, v)$ has been defined in Eq. 2.

B. Constructing Feature Vectors

Algorithm 1 Process of hiding private information.

Input: Feature matrix $F \in \mathbb{R}^{n\times x}$; Adjacency matrix $A \in \mathbb{R}^{n\times n}$; Node set V; Edge set E; The set of neighbors of each node $N(\cdot)$; The set of degree lists of each node $deList[\cdot]$; The distance matrix of nodes $Distance \in R^{n\times n}$; The similarity matrix of nodes $S \in R^{n\times n}$.

Output: Similarity matrix $S \in \mathbb{R}^{n\times n}$; New feature matrix $nF \in \mathbb{R}^{n\times d}$.

$Distance = NULL$, $S = NULL$;
for each $u \in V$ **do**
  $deList[u] = NULL$;
  for each $v \in N(u)$ **do**
    Computing d_v according to Eq. 1;
    $deList[u].append(d_v)$;
  end for
  $deList[u].sort()$;
end for
for $u, v \in V$ **do**
  Computing $Distance[u][v]$ according to Eq. 2 and 3;
  $S[u][v] = e^{-Distance[u][v]}$;
end for
$nF = one - hot\_encoding(F)$;
return S, nF;

Data from different users may include name, age, and more private information. The useful data for machine learning models should be protected carefully. The process of hiding information should happen before training the models. In our work, content features are processed using one-hot encoding. One-hot encoding is a method to map features into a group of bits among which the standard combinations of values are only those with a single 1 bit and all the others 0 bits. Models receive messages from the one-hot encoding matrix. At this point, we complete hiding users' privacy by redefining the relation among nodes. Algorithm 1 describes the process of hiding the private information.

C. Graph Neural Networks

GCNs can be considered as a simplification of the traditional graph spectral methods. For graph $G = (V, E)$, the normalized Laplacian matrix is denoted as $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^{\top}$, where D is the degree matrix and I is an identity matrix. U is a matrix which consists of eigen-vectors of matrix L. The spectral operator is used to encode the structure and features of nodes. However, for large-scale graphs, the computation will become costly. To solve this problem, different approximations are proposed to reduce the computation of both matrix L and the spectral convolution. The representations of nodes convolutional features in k^{th} layer can be computed as Eq. 5.

$$h_v^k = \sigma(\Sigma_{u\in\{v\}\cup N(v)}\hat{a}_{vu}W^k h_u^{k-1}) \tag{5}$$

where $\hat{a}_{vu}$ is an element of re-normalization of the adjacency matrix Adj, and $N(v)$ denotes the set of node v's neighbors. W^k is a trainable matrix of k^{th} layer. h_v^k is the representation of node v in k^{th} layer and h_u^{k-1} denotes the representation of node in $k-1^{th}$ layer. In the input layer, $h_u^0 = F_u$, where F_u is the row of node u in the feature matrix.

However, using convolutional layer in the neural network may lose information when mapping vectors into low-dimension space. To make up to the loss of information, we

add dense layers into the original model. In convolutional layers, information from neighbors is filtered to a vector in low-dimension space. It expresses that some information would be lost. To reduce the loss of information, we use several dense layers after the input convolutional layer. We use ReLU function as the activation function of the neural cell. The function ReLU is shown in Eq. 6. In dense layers, neural cells are fully connected to others. We use handled content information and structural information as the input to predict the improved GCN model. The training process is shown in Algorithm 2.

$$ReLU(x) = max(0, x) \tag{6}$$

Algorithm 2 The process of training.

Input: Similarity matrix S; New feature matrix nF; Label matrix L;
Output: Node representation matrix R;
1: **for** $epoch \in eopches$ **do**
2: Updating the parameters according to Equation 5;
3: **end for**
4: **return** R;

IV. Experiments

To evaluate SGNN on node classification tasks, we conduct experiments on SGNN and three state-of-the-art models, GCN [7], Struc2vec [6] and DEMO-Net [10], with the same experimental settings. The data sets of experiments consist of Aminer [22], Brazil and Europe which were used in [6].

A. Data sets

- Aminer[1]. Aminer [22] is a data set of academic network. It includes paper information, paper citation and author collaboration. $2,092,356$ papers, $8,024,869$ citations, $1,712,433$ authors and $4,258,615$ collaboration relationships are saved in the data set. To prepare for the node classification, we extract 3923 authors from 8 categories[2] of conferences in Google Scholar. The process of dealing with the data set is the same as Metapath2vec [14]. The data set is a sparse graph. We use authors as nodes of graph and the collaborations of authors are captured as the edges among nodes. The features of authors in Aminer are used to present the original content information of nodes. The collaborations of authors are used to present the original structural information of the network.
- Brazil[3] and Europe[4]. Brazil and Europe are traffic flights of airports [6]. The Brazil data set has 131 nodes and 1038 edges among nodes. Every node denotes an airport and every edge denotes an flight between two airports. The label of each airport denotes the level of the airport. The Europe data set is also an airports data set which is similar to Brazil. It includes 399 nodes and 5995 edges in the network. Compared with Aminer, these two data sets are considered as dense graphs. In the two data sets, we use the flights among airports to present the structural information, and use the identity matrix to present the content information.

The detailed information of the data sets are shown in Table I.

TABLE I
Detail information of Datasets: Aminer, Brazil and Europe

Dataset	Nodes	Edges	Classes
Aminer	3923	9023	3
Brazil	131	1038	4
Europe	399	5995	4

B. Experiments Compared With Existing Models

1) Comparative experiments using adjacency matrix: To compare with GCN, Struc2vec and DEMO-Net fairly, we use the same setting of parameters with their open codes. Then we conduct experiments to compare the classification performances of SGNN and the other three models on the three data sets, i.e., Aminer, Brazil and Europe data sets.

The result of experiments on data sets using adjacency matrix is shown in Tabel II. We conclude the experiment results from the following aspects:

- In experiments on the same data set, SGNN performs better than GCN, Struc2vec and DEMO-Net. Detailedly, in experiments on Aminer, SGNN improves accuracy by about 3% compared with GCN and 21.2% compared with DEMO-Net. The experiment of Struc2vec on Aminer costs so much time that no result is obtained at last. In experiments on Brazil and Europe, SGNN gets better results than that of GCN and Struc2vec but it gets lower classification accuracy than that of DEMO-Net. The reason why DEMO-Net performs better than SGNN is that Demo-Net captures structure with a strong condition. So DEMO-Net shows better performance on dense graphs in which structure information are luxuriant. Results show that SGNN performs better than these existing methods on large-scale sparse graphs.
- In experiments on the three different data sets, SGNN shows different performance on classification tasks. In experiment on Aminer data set, SGNN obtains a 81.4% accuracy which much higher than that on Brazil and Europe. As is mentioned in IV-A section, Aminer is a large-scale graph but Brazil and Europe are small-scale graphs. We consider the reason why SGNN underperforms Struc2vec under the Europe data set is that the structural units in small-scale are densely distributed and learned overly. While, the structural units in Aminer

[1]https://www.aminer.cn/aminernetwork

[2]Computational Linguistics, Computer Graphics, Computer Networks & Wireless Communication, Computer Vision & Pattern Recognition, Computing Systems, Databases & Information Systems, Human Computer Interaction, and Theoretical Computer Science.

[3]The data set is collected from ANAC which can be visited at http://www.anac.gov.br/.

[4]The data set is collected from Eurostat which can be visited at http://ec.europa.eu/.

which is considered as the large-scale graph are sparsely distributed and captured properly. The results demonstrate that SGNN is more suitable for classification on large-scale graphs.

TABLE II
THE RESULTS OF NODE CLASSIFICATION ON ADJACENCY MATRIX OF AMINER, BRAZIL AND EUROPE.

	SGNN	GCN	Struc2vec	DEMO-Net
Aminer(Adjacency)	**81.4%**	78.4%	-	60.2%
Brazil(Adjacency)	**28.0%**	28.0%	14.3%	61.4%
Europe(Adjacency)	**18.8%**	12.1%	27.5%	47.9%

Experiments using adjacency matrix of original graph above will lead to the leakage of information. So we conduct experiments using similarity matrix described in section III-A with the same other experimental conditions.

2) Comparative experiments using similarity matrix: We compare the performance of SGNN and GCN on this set of experiments, because Struc2vec and DEMO-Net are designed suing the topological structure without weights on the edges rather than weighted structure. The result of experiments on data sets using similarity matrix is shown in Table III. We conclude the result as follows:

- In experiments on Aminer, SGNN outperforms GCN with accuracy of 2.7%. It also performs better than GCN on Europe data set with about 1%. The comparison of SGNN and GCN is shown in Fig. 4. We analyse the reason why SGNN outperforms GCN is that SGNN preserves more nodes' information when training parameters among layers.
- For experiments of SGNN on different data sets, SGNN gets an accuracy of 71.8% on Aminer, 18.0% on Brazil and 21.5% on Europe. As is obviously shown in Fig. 4, SGNN performs much better on Aminer than that on Brazil and Europe. The reason is that neural networks in SGNN do well in capturing high-distance structure and aggregating numerous information of nodes in the large-scale sparse graph. Therefore SGNN is more suitable to solve problems on large sparse networks.
- The experiments on data sets using similarity matrix hide the original structure information from data sources and protect users' privacy prudently.

TABLE III
THE RESULTS OF NODE CLASSIFICATION ON SIMILARITY MATRIX OF AMINER, BRAZIL AND EUROPE.

	SGNN	GCN
Aminer(Similarity)	**71.8%**	69.1%
Brazil(Similarity)	**18.0%**	18.0%
Europe(Similarity)	**21.5%**	20.8%

C. Experiment on Aminer

In this section, we introduce the experiment of SGNN on Aminer and analyse the evaluation index of the model. To

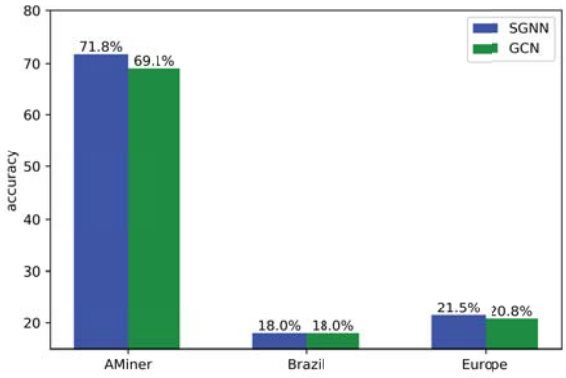

Fig. 4. The Comparison of SGNN and GCN in the experiments on data sets with similarity matrix.

complete the protection of original information on Aminer, we hide the users' features using one-hot encoding and hide the structural information by methods mentioned in III section. We divide the Aminer data set into train set, validation set and test set. The train set learns the parameters of the model, and the validation set evaluates the trained model. Then the model is improved by updating the parameters. The test set is used to get the accuracy of the node classification task.

The relation of loss function and epoches is shown in Fig. 5. As is revealed in Fig. 5, the value of the loss function is about 1.09. With the epoch increasing, the loss of train set is decreasing continuously to the value about 0.91. While the loss on validation set decreases to about 0.89. It proves that the optimization is effective.

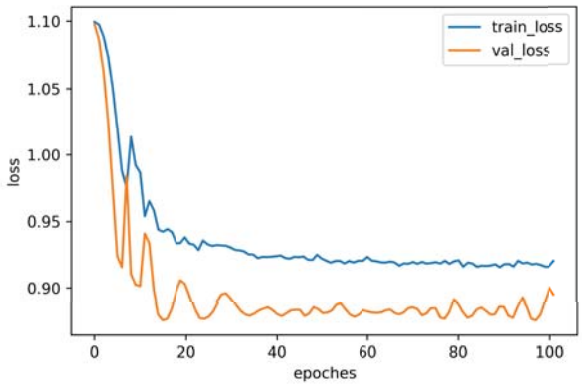

Fig. 5. The value of the loss function on train set and validation set.

With the progress of training, the accuracy of node classification is shown in Fig. 6. At the beginning of training, the accuracy on train set is about 31.1%. While, the accuracy on train set is increasing with the epoches increasing. At the end of training, the accuracy on train set is about 59.4%. The accuracy on validation set is also shown on the Fig. 6. We can see that the accuracy on validation set is higher than that

on train set. The final accuracy is about 64.6%. On test set, the accuracy is about 71.8%. The accuracy shows that SGNN gets superior performance on node classification tasks.

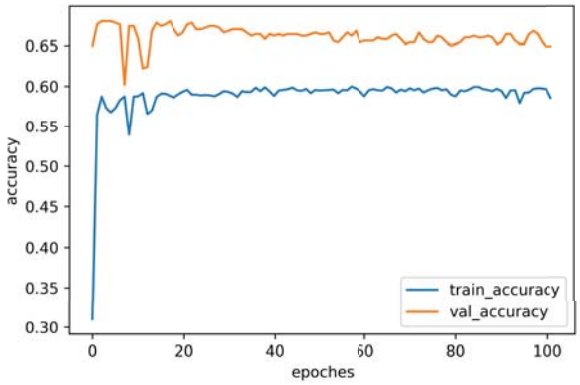

Fig. 6. The accuracy on train set and validation set.

D. Discussion

The results of experiments demonstrate the superior performances of SGNN on node classification. We discuss the several experiments from the following aspects:

- The experiments are taken under the condition that original information is hidden. The features of nodes are encoded by one-hot encoding. The similarity of nodes are computed to take place of adjacency matrix. The neural network exchanges messages with handled data information instead of original data sources. However, traditional methods use original graph information to train models to complete various tasks. The way of data preprocessing achieves the purpose of federated learning which is to protect users' privacy.
- As is seen in experiment result, the accuracy of SGNN on node classification are higher than GCN. GCN uses its convolutional layers to filter node information. While, this may cause the loss of information. SGNN uses hidden dense layers to reduce the loss of information. Moreover, the distance of a pair of nodes is redefined to describe the relation of nodes more precisely. Compared with traditional methods that regards neighbors of a node equally, our method shows superior performance on classification tasks.
- Compared with GCN, SGNN improves the classification accuracy on Aminer. What is worth taking note of is that we also take experiments on Struc2vec and DEMO-Net, and the results show that SGNN can performs well on the Aminer, which is considered as the large-scale grah. However, they do not get a satisfactory result on the large graphs. Aminer is a sparse graph in which there are a large number of nodes with a small number of edges, while Brazil and Europe are two dense graphs. The graph neural networks aggregate information from nodes in a long distance from current node through hidden layers. While, Struc2vec and DEMO-Net fail to capture information of nodes far away from each other.

V. Conclusion

Many models have been proposed to deal with node classification problems. Content features and node structure should be well considered to classify the nodes with different labels. Moreover, users' information should be well protected.

To this end, We take advantage of federating learning and propose SGNN for classifying the nodes in network. The content information is hidden by one-hot encoding. The structural information is hidden by computing the similarity of nodes. The information from different data sources are delivered to the collaborator SGNN. SGNN trains parameters and learns nodes representations to complete node classification tasks. Experiment results show that SGNN improves the classification accuracy than existing models.

To conclude, SGNN improving over existing approaches in several aspects:

1) It captures the structural information by redefining the distance between a pair of nodes, and computing the similarity of nodes. The similarity matrix takes the place of the adjacency matrix to describe the relations of nodes more precisely.
2) It takes the advantage of federated learning to protect the users' private information.
3) It can learn node representation from large-scale graph and perform better than existing models.

There are also some improvements of our proposed model, that may be addressed in the future work. Firstly, SGNN can be improved to handle the classification on the small-scale networks. Secondly, the way of protecting users' privacy should be more carefully designed. In addition, the model is likely to be improved to solve the graph classification problems.

VI. Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61872222, the Key Research and Development Program of Shandong Province under Grant 2017CXGC0605, 2017CXGC0604, 2018GGX101019, and the Young Scholars Program of Shandong University.

References

[1] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.
[2] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of mathematical sociology*, vol. 1, no. 1, pp. 49–80, 1971.
[3] L. D. Sailer, "Structural equivalence: Meaning and definition, computation and application," *Social Networks*, vol. 1, no. 1, pp. 73–90, 1978.
[4] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[5] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, 2015, pp. 891–900.

[6] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 385–394.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[9] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[10] J. Wu, J. He, and J. Xu, "Net: Degree-specific graph neural networks for node and graph classification," *arXiv preprint arXiv:1906.02319*, 2019.

[11] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[13] B. Perozzi, R. Al-Rfou, and 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.

[14] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2017, pp. 135–144.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," *arXiv preprint arXiv:1906.04817*, 2019.

[17] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," *arXiv preprint arXiv:1903.11960*, 2019.

[18] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.

[19] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *arXiv preprint arXiv:1812.03337*, 2018.

[20] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "Secureboost: A lossless federated learning framework," *arXiv preprint arXiv:1901.08755*, 2019.

[21] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 7, no. 3, p. 10, 2013.

[22] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 990–998.