Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 5702519, 9 pages
https://doi.org/10.1155/2020/5702519

Research Article

Enhanced Unsupervised Graph Embedding via Hierarchical Graph Convolution Network

H. Zhang,[1] **J. J. Zhou**,[1] **and R. Li**[1,2]

[1]*School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China*
[2]*Key Laboratory of Computation Intelligence and Chinese Information Processing of Ministry of Education, Shanxi University, Taiyuan 030006, China*

Correspondence should be addressed to H. Zhang; zhanghu@sxu.edu.cn

Received 14 May 2020; Accepted 19 June 2020; Published 26 July 2020

Guest Editor: Shianghau Wu

Graph embedding aims to learn the low-dimensional representation of nodes in the network, which has been paid more and more attention in many graph-based tasks recently. Graph Convolution Network (GCN) is a typical deep semisupervised graph embedding model, which can acquire node representation from the complex network. However, GCN usually needs to use a lot of labeled data and additional expressive features in the graph embedding learning process, so the model cannot be effectively applied to undirected graphs with only network structure information. In this paper, we propose a novel unsupervised graph embedding method via hierarchical graph convolution network (HGCN). Firstly, HGCN builds the initial node embedding and pseudo-labels for the undirected graphs, and then further uses GCNs to learn the node embedding and update labels, finally combines HGCN output representation with the initial embedding to get the graph embedding. Furthermore, we improve the model to match the different undirected networks according to the number of network node label types. Comprehensive experiments demonstrate that our proposed HGCN and HGCN* can significantly enhance the performance of the node classification task.

1. Introduction

Graph embedding aims to map high-dimensional sparse network data to low-dimensional dense real-value vector space, which can adaptively extract features and be applied to the network analysis tasks, such as node classification, link prediction, and dimensional visualization [1, 2]. Generally, graph embedding models mainly include the shallow embedding model and the deep embedding model. The shallow embedding model represented by Deepwalk is directly learned by defining the loss function for network representation, which is usually an unsupervised learning method [3]. This model is suitable for the undirected graph, but it also has the disadvantage of ignoring the feature information in graphs. On the contrary, the deep embedding model represented by the GCNs uses the additional feature and label information of the graph at the same time, which is generally a semi-supervised or supervised learning method. However, the deep embedding model is not more suitable for undirected graphs with only structure information, at the same time many real networks do not include the node features and labels due to privacy protection and annotation difficulty, so the study of deep unsupervised graph embedding method for a real undirected graph is urgent.

To help address these issues, we propose a novel unsupervised graph embedding method with a hierarchical graph convolution network, which does not need the extra features and labeled data. The general process is shown in Figure 1, which includes 3 layers with the initial, update, and output layer. Among them, the input of the model is a graph, the output is the low-dimensional vector representation of nodes that relate to the number of label types, and the detailed structure of the model is partly omitted.

Our contributions are the following: (1) we propose an enhanced hierarchical graph convolution network for the undirected graph with only structure information. (2) According to the number of label types in the different experimental data sets, we introduce max-pooling to

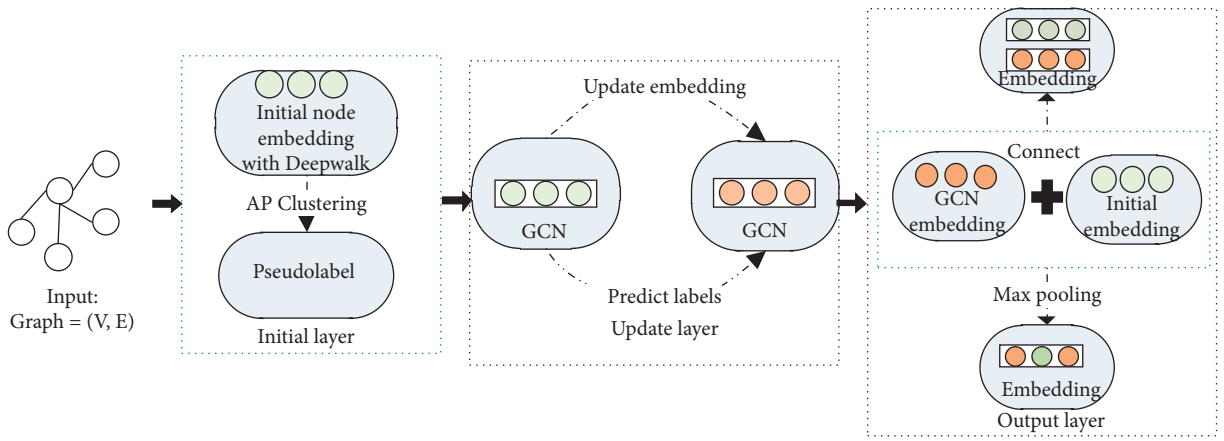

Figure 1: Model workflow. The model includes 3 layers with the initial, update, and output layer. In the output layer, the proposed model is improved according to the number of label types.

improve the proposed model. (3) We conduct the node classification experiments on Wikipedia, American air-traffic, Cora, and Citeseer data sets separately, and our proposed method achieves significant improvements than other state-of-the-art baselines.

2. Related Work

Graph embedding methods have different classification systems. In this section, firstly, we introduce the traditional graph embedding methods. Then, we review the different graph embedding methods in the field of machine learning. In the end, we classify the graph embedding models according to the structure of the embedding methods, and a profound analysis of the disadvantages of the existing graph embedding methods is given. Additionally, we further introduce existing GCN variants for graphs.

2.1. Traditional Graph Embedding Methods. Traditional network representation learning methods are realized by dimension reduction technology [4]. Classic dimension reduction methods include Principal Component Analysis (PCA) [5] and Multi-dimensional Scale (MDS) [6]. Both methods can capture linear structure information, but they cannot acquire a nonlinear structure in input data.

From a linear algebraic perspective, all unsupervised graph embedding methods are generally represented by the various graph matrices, especially the Laplace operator and the adjacent matrix [7]. In terms of computational efficiency, the feature decomposition of the data matrix is expensive.

2.2. Graph Embedding Methods for Machine Learning. In the representation learning for machine learning, the semi-supervised graph embedding method requires a set of features that can distinguish nodes [8]. Typical model mainly uses manual feature extraction to learn node representation for specific professional fields, which has the disadvantages of the inaccuracy and poor robustness. Another method mainly learns node representation such as MMDW [9] by solving optimization problems, which improves the accuracy of the extracted feature. However, the number of estimation parameters is large and the time complexity is high [10].

As for the unsupervised graph embedding method, to balance accuracy and computational efficiency, the feature representation needs to define an objective function independent of downstream tasks, such as Deepwalk and LINE. The above unsupervised graph embedding method only uses network structure information to obtain low-dimensional embedding. Moreover, the graph embedding method also learns from the rich content of node attribute and edge attribute. TADW [11] incorporates text features of vertices into network representation learning under the framework of matrix factorization. CENE [12] integrates text modelling and structure modelling in a general framework by treating the content information as a special kind of node. HSCA [13] is a network embedding method of attribution graph, which simulates homophone, network topology, and node characteristics at the same time. Inspired by pre-training methods, pre-training models based on GCN are proposed. DeepGraphInfoMax [14] relies on maximizing mutual information between patch representations and corresponding high-level summaries of graphs, while both of them derived using established graph convolutional network architectures. The pre-trained GCN model proposed by Hu et al. [15] can capture generic graph structural information that is transferable across tasks.

Compared with the methods of manually extracting features, the representation method by optimizing objective function can obtain more comprehensive features, which is closely related to the downstream predicting task [16]. Therefore, the unsupervised graph embedding method improves the disadvantages of semi-supervised machine learning method, such as difficult extensibility and high training complexity.

2.3. Graph Embedding Methods with Different Structures. According to the structure of the representation methods [17], graph embedding methods are divided into shallow embedding method and deep embedding method. In the

shallow embedding model, inspired by the successful application of skip-gram [18] in natural language processing [19], a series of skip-gram based models are proposed to encode network structures into continuous spatial vector representations in recent years, such as Deepwalk, LINE, and Node2vec.

Deepwalk [20] simulates word sequences to generate node sequences by random walk from each node, which forms a "corpus" based on those node sequences. Furthermore, Deepwalk sets the size of the background window and then imports the "corpus" into the skip-gram model to get the node representation. The first-order similarity of direct connection and the second-order similarity of shared neighbor nodes are optimized respectively, and the two similarities are combined at the output of the nodes in LINE [21]. Node2vec [22] uses two additional parameters to control the direction of the random walk on the generation step of the "corpus" of Deepwalk. Struc2vec [23] defines the nodes that are not structurally adjacent but have the same structural roles. The shallow embedding models usually use unsupervised machine learning methods, so they have the disadvantage of ignoring the node features.

In the deep embedding model, the mainstream methods use the deep learning model to capture the nonlinear relationship between nodes [24]. Typical deep learning models are NE-FLGC, SDNE, GCN, and a series of GCN variants.

NE-FLGC [25] studies the problem of representation learning for network with textual information, which aims to learn low-dimensional vectors for nodes by leveraging network structure and textual information. SDNE [26] automatically captures the local relationship of the nodes by using the unsupervised learning method and takes the second-order neighbor of the nodes as the input to learn the low-dimensional representation of graphs, but the model still does not consider the node features.

GCN [27] is a deep semi-supervised graph embedding model with incorporating the extra feature and labeled data into the graph embedding learning process, which cannot be easily applied to the undirected graph with only structure information. In terms of applicability, the existing GCN variants are mainly studied in two aspects. One variant assumes the graph is attributed, such as GraphSAGE [28], GAT [29], N-GCN [30], and Fast-GCN [31]. GraphSAGE [28] can be used to generate node embeddings for previously unseen nodes or entirely new input graphs, as long as these graphs have the same attribute schema as the training data. GAT [29] uses attention mechanism to address the shortcomings of prior methods based on graph convolutions or their approximations. N-GCN [30] improves the scalability of GCN on the whole graph by setting the size of the convolution kernel. Fast-GCN [31] is a batch training algorithm combining importance sampling, and it can not only make GCN training more efficient but also generalize well for inference. Generally, they take the structure information or the node degree information as the feature and then use the correct label to train GCN to learn the graph embedding. However, the model still needs to incorporate the correct labeled data into the graph embedding processing. Another variant is proposed to solve the label problem, such as M3S and GMNN. M3S [32] uses the correct extra feature and enlarges the labeled data by self-training to learn the undirected graph embedding. GMNN [33] similarly applies the correct extra feature and neighbor nodes to generate pseudo-labels for unsupervised learning. As a consequence, the previous method still cannot consider the feature and the label of the node at the same time.

3. Model

In this section, we describe the proposed method and give a detailed framework for the model. Firstly, we generate the initial node embedding using Deepwalk. Simultaneously, the pseudo-labels are built by the AP clustering algorithm [34]. Then, we build the hierarchical GCN to update the node embedding and labels. Finally, we introduce max-pooling to enhance our model according to the number of label types in the different experimental datasets.

3.1. Initial Node Embedding. The first step of the model is to generate the initial node embedding using Deepwalk. Deepwalk is the shallow embedding model with random walking, and it can better learn the global structure and the context of the node in the undirected graph. Therefore, Deepwalk is used as a pre-training method to generate the initial node embedding in this paper.

Let $G = (V, E)$ be a given undirected graph with vertex set V and edge set E, where $n = |V|$ denotes the number of nodes in the graphs. Our formal definition is general and can apply to any undirected and unweighted network. Firstly, we acquire a specific length random walk sequence $w(v_i)$ for the node v_i, which is shown in the following formula (1):

$$w(v_i) = \left(w(v_i)^1, w(v_i)^2, \ldots, w(v_i)^k\right), \tag{1}$$

where $w(v_i)^n$ is the n^{th} node. And then, we use a language model to learn from these random walk sequences and denote the initial embedding as follows:

$$E_M(w(v_i)) = \text{skip} - \text{gram}(w_i^u), \tag{2}$$

where w_i^u is the word sequence in the language model, and skip – gram is the language model. $E_M(.)$ is denoted as the output function. We use $w_i^u = (w_1, w_2, \ldots w_n)$ to denote the word sequence, where w_i is a specific word in the language model. The detailed processing flow is shown in Figure 2.

3.2. Pseudolabels. The second step of the model is to construct the pseudo-labels and predict labels with GCN. After obtaining the initial node embedding of the undirected graph, we use the clustering method to acquire pseudo-label y_i for the unlabeled node i, where $y_i \in Y$ and Y is a pseudo-label set. The specific processing flow is shown in Figure 3.

And now, we update G to denote the graph, as shown in the following formula (3):

$$G = (V, E, E_M(w(v_i)), Y). \tag{3}$$

In this paper, AP clustering model called near-neighbor propagation clustering algorithm is adopted to generate

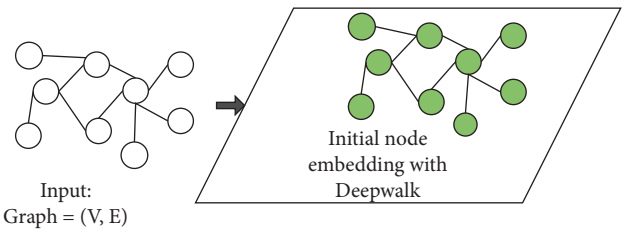

Figure 2: The process of generating the initial node embedding. The input of the model is an undirected graph.

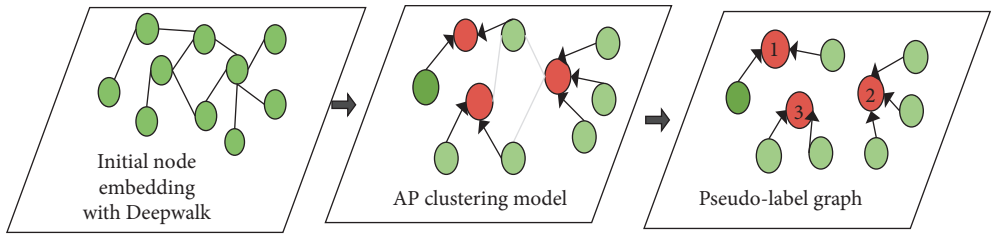

Figure 3: The processes of generating pseudo-labels with the AP clustering method.

pseudo-labels, which is mainly because the method contains the following three advantages. (1) Because the number of label categories is unknown, the model needs to choose a clustering method that does not need to specify the number of clusters. Unlike k-means and k-center algorithms, AP clustering model is not necessary to set the final number of clusters when clustering. (2) The cluster centers of AP algorithm are the actual nodes in the data set, which are the representative of each class, so AP clustering model is not sensitive to the initial negative value of the node embedding. (3) If the sum of squares of errors is used to measure the performance of the algorithms, the sum of squares of errors of AP clustering is lower than that of other methods. This evaluation index shows that AP clustering method is effective.

Based on this, we further use GCN to predict the label for the unlabeled node and define the predicted maximum value as the label y_i', where $y_i' \in Y'$ and Y' is the predication label set. And now, formula (3) is updated as following in formula (4):

$$G = \left(V, E, E_M\left(w\left(v_i\right)\right), Y'\right). \tag{4}$$

3.3. Hierarchical GCN. In this section, we focus on the hierarchical GCN model, which includes two GCN [27] models. At the start, we introduce GCN. Next, we set up hierarchical graph convolution network to propose the HGCN model and the HGCN* model by analyzing the updated undirected graphs. Additionally, we further explain the reason for the improved model.

3.3.1. Graph Convolution Network. Different from the conventional convolutional neural networks (CNN), the convolution operation for a graph is defined as the weighted average of neighbors of one particular node. Mathematically, the graph convolution layer is defined as

$$H^{(l+1)} = \sigma\left(PH^{(l)}W^{(l)}\right), \tag{5}$$

where $P = D^{-1/2}(A + I)D^{-1/2}$ is the normalized Laplacian matrix of the graph G. $H^{(l)}$ is the input of the l^{th} hidden layer of GCN, i.e., the output of the $(l-1)^{\text{th}}$ hidden layer. $W^{(l)}$ is the weight matrix in the l^{th} hidden layer that would be trained. A is the adjacency matrix of the undirected graph. And $\sigma(\cdot)$ is the activation function. For any given node, a GCN layer aggregates the previous layer's embedding of its neighbor with A, followed by linear transformation W and nonlinear activation σ, so as to obtain a contextualized node representation. We denote $F_w(\cdot)$ as L-layer GCNs, parametrized by $\{w_i\}_{i=1}^{L}$. For each given graph G = (V, A) with input features $H^{(0)}$, $F(\cdot)$ can get node representations by:

$$F_w(G) = \sigma\left(P\left(..\sigma\left(PH^{(1)}W^{(1)}\right)..\right)W^{(L)}\right). \tag{6}$$

3.3.2. HGCN and HGCN.* After the initial node embedding section and the pseudo-labels section, we have learned the graph information from formula (4). After predicting labels, we also update the graph embedding with GCN, which is as shown in formula (7):

$$G = \left(V, E, E_M'\left(w_{v_i}\right), Y'\right), \tag{7}$$

where $E_M'(w_{v_i})$ is the updated node embedding by the first GCN.

According to formula (6), $E_M'(w_{v_i})$ is defined as formula (8):

$$E_M'\left(w_{v_i}\right) = \sigma\left(P\left(..\sigma\left(PE_M\left(w\left(v_i\right)\right)W^{(0)}\right)..\right)W^{(2)}\right), \tag{8}$$

where $H^{(0)} = E_M(w(v_i))$ is the input feature, which is the initial node embedding using Deepwalk. In this paper, we

use 3-layer GCNs with 128-dimensional hidden vectors to accomplish the pre-training and adaptation.

And then, we further use the second GCN to learn the undirected graph again. The final embedding $E''_M(w_{v_i})$ is denoted as formula (9):

$$E''_M\left(w_{v_i}\right) = \sigma\left(P\left(..\sigma\left(PE'_M\left(w_{v_i}\right)W^{(0)}\right)..\right)W^{(2)}\right). \tag{9}$$

Moreover, after this second learning process, GCN will lead to too smooth for the embedding, which can affect node classification. Therefore, to relieve smoothing, we combine the secondary learning representation $E''_M(w_{v_i})$ with the initial node embedding $E_M(w_{v_i})$. Finally, we get the ultimate embedding $O(G)$ for the undirected graph as shown in formula (10):

$$O(G) = E''_M\left(w_{v_i}\right) \oplus E_M\left(w_{v_i}\right), \tag{10}$$

where $\oplus$ is a vector splicing operation.

The previous HGCN method can be applied to learn a better embedding, when the graph is an undirected graph with a few numbers of node label types. However, when the undirected graph has a large number of node label types, the proposed model will also decompose labels into multi-dimensional features and add them to the learning processing, which can lead to the feature redundancy. Therefore, motivated by this analysis, we introduce max-pooling to improve the embedding model for the graph with a large number of label types, namely HGCN*. The exact processing flow is shown in Figure 4.

4. Experiments

4.1. Datasets. To verify the effectiveness of the proposed model, we conduct several comparative experiments on four open available graph datasets:

(i) Cora dataset [35] consists of 2,708 machine learning papers classified into one of seven classes and 5,429 links between them.

(ii) Citeseer dataset [35] is a link dataset built with permission from the Citeseer web database. It contains 3,327 publications from six classes and 4,732 links among them.

(iii) Wikipedia dataset [36] consists of 2,405 Wikipedia pages from 17 categories and 17,981 links between them. It is much denser than Citeseer and Cora datasets.

(iv) American air-traffic dataset [23] is from the United States Air Traffic Network. It contains 1190 airports and 13599 relationships between airports and airports, and the airport labels are divided into four groups.

All specific nodes and edges in our experimental datasets are shown in Table 1.

Note that all datasets are used as undirected graphs with only network structure information and the correct labels in the datasets are only used in the node classification verification task in this paper.

4.2. Baselines. For comparison, we adopt six kinds of baselines as follows:

(i) Deepwalk [20] is a typical unsupervised graph embedding method which adopts the skip-gram language model.

(ii) LINE [21] is also a popular unsupervised method which considers the first-order and second-order proximity information.

(iii) Node2vec [22] learns low-dimensional representations for nodes in a graph by optimizing a neighborhood preserving objective, which explores the structure of the network by controlling the p and q parameters, and learns the d dimension feature representation by simulating biased random walks.

(iv) GraRep [37] captures the long-distance nodes and uses the matrix decomposition method to learn the d dimension feature representation of the node.

(v) Hope [38] introduces higher-order similarity and uses the matrix decomposition method to learn the d dimension feature representation of the node.

(vi) SDNE [26] learns the d dimension node representation by capturing the network nonlinear structure using multi-layer nonlinear functions, which is a semi-supervised deep graph embedding model.

Several existing methods include shallow embedding models, matrix decomposition models, and deep embedding models. Note that since some existing work ([27–29]) utilizes extra node features, we do not compare with them directly.

4.3. Node Classification Experiments. To verify the effectiveness, we follow the same settings used in Hamilton et al. (2016) [27], including the following benchmark tasks: (1) using HGCN to classify nodes on American air-traffic, Cora, and Citeseer datasets; (2) using HGCN* to classify nodes on Wikipedia dataset.

4.3.1. Experimental Settings. In this experiment, Micro-F1, Macro-F1, and Accuracy are used as measurements. The model uses the same proportion of training and test data as the existing published papers, i.e., 50% of the training data and 50% of the test data.

To facilitate the comparison, on the Cora and Citeseer datasets, we report the results of one experiment for comparison with the baseline models. However, on the American air-traffic network dataset, we repeat the experiment 5 times using random samples for training and demonstrate the average performance. The experimental results are given in Table 2. "*" denotes the published results of the existing papers [29], and the roughened numbers represent the best results.

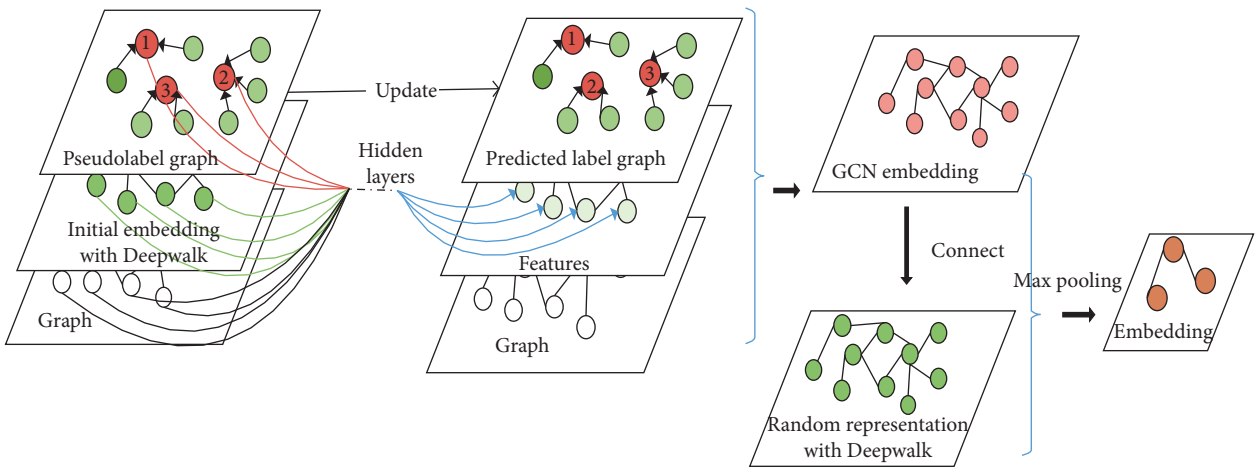

Figure 4: HGCN*. When the undirected graph node has a large number of label types, the model selects features with max pooling in the output layer.

Table 1: Datasets.

Datasets	Cora	Citeseer	Wikipedia	American air-traffic
Nodes	2708	3327	2405	1190
Edges	5429	4732	17981	13599
Classes	7	6	17	4

Table 2: Node classification results with a few label types.

Datasets	American air-traffic			Cora			Citeseer		
Evaluating indicator	Micro-F1	Macro-F1	Accuracy	Micro-F1	Macro-F1	Accuracy	Micro-F1	Macro-F1	Accuracy
Deepwalk	0.523	0.518	0.523	—	—	0.672*	—	—	0.432*
LINE(2nd)	0.504	0.499	0.504	0.704	0.684	0.704	0.444	0.399	0.444
Node2vec	0.493	0.459	0.493	0.771	0.765	0.771	0.567	0.486	0.567
SDNE	**0.580**	0.560	**0.580**	0.660	0.648	0.660	0.400	0.351	0.400
HGCN	0.571	**0.568**	0.571	**0.797**	**0.780**	**0.797**	**0.568**	**0.518**	**0.568**

On the Wikipedia dataset, we use the same measurements to compare with the results of open-source framework OpenNE [39]. The best experimental results are presented in Table 3.

4.3.2. Results and Discussion. As shown in Table 2, it can be seen that our proposed model achieves the best results compared to all shallow baseline models, and the outcomes are very close to the deep baseline model SDNE. On the Cora and Citeseer datasets, the experimental results of our model are higher than all baseline models in all measurements, which show that the model can effectively learn the undirected graphs.

Obviously, on the American air-traffic dataset, the performances of HGCN are close to the results of SDNE. The main reason is that the nodes on American air-traffic dataset are labeled by the network global structure. In other words, the node label of the dataset is more related to their structure identity than the labels of their neighbors. HGCN uses the label information of neighbor nodes to update and learn the labels for the unlabeled data, so the node classification performances of HGCN are significantly affected. To visually display the performances of HGCN, the accuracy is presented in Figure 5.

As illustrated in Figure 5, we compared the accuracy of all baselines including the shallow and the deep embedding model on the three undirected graph datasets. The average performances of the shallow embedding model are higher than SDNE model on the Cora dataset and the Citeseer dataset. And the result of the SDNE is better than the other shallow baselines on the American air-traffic dataset. This suggests that SDNE can learn well the special structure network, which node labels are related according to the network global structure. While the proposed model HGCN has a significant increase on most experimental datasets, which further verifies the applicability of our model on undirected graphs.

As for the improved model according to the number of label types, we verify its effectiveness on the Wikipedia dataset and compare it with the results of the open-source framework OpenNE [39]. The experimental results are shown in Table 3.

According to the results of Table 3, when the HGCN model uses only vector splicing, the performance degrades in Micro-F1. In Macro-F1, the results are better than the other all baseline models except for Deepwalk. That suggests that

Table 3: Node classification results on Wikipedia.

	Baselines	Micro-F1	Macro-F1
Random walk	Deepwalk	0.669*	0.560*
	LINE(2nd)	0.576*	0.387*
	Node2vec	0.651*	0.541*
Matrix decomposition	GraRep	0.633*	0.476*
	Hope	0.601*	0.438*
Deep model	SDNE	0.643*	0.498*
Our model	HGCN	**0.599**	**0.549**
	HGCN*	**0.717**	**0.590**

The result with * is from the open-source OpenNE [39].

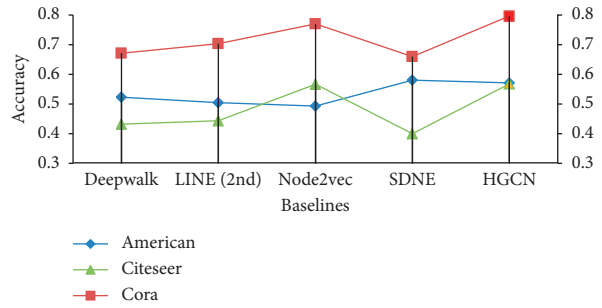

Figure 5: Node classification performances (accuracy) with fewer label types.

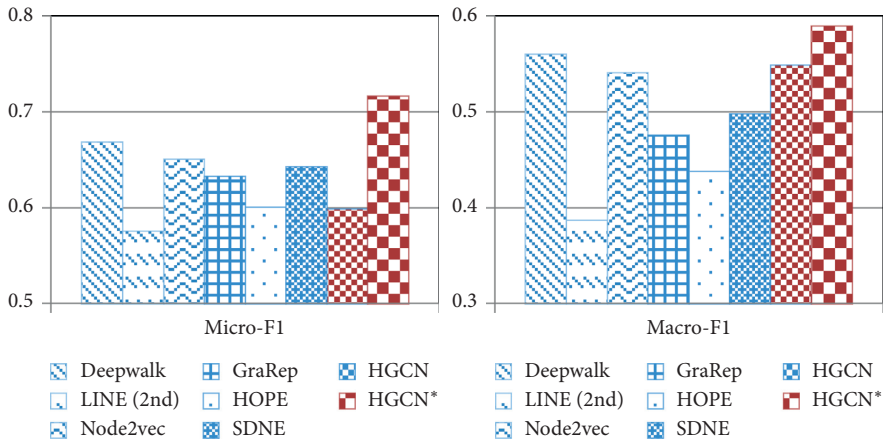

Figure 6: Node classification performances (Micro-F1; Macro-F1) with baseline models on the Wikipedia dataset.

the HGCN model breaks up pseudo-labels into the features, while the Wikipedia dataset leads to features redundant because of more label types.

To help address this problem, the max-pooling layer is introduced to the HGCN model, which can select the features to improve the effect of node classification on the Wikipedia dataset. As can be seen from the experimental results in Table 3, compared to Deepwalk, LINE (2nd), and Node2vec, the Micro-F1 of HGCN* is improved by 4.8%, 14.1%, and 6.6%. And in Macro-F1, the results are enhanced by 3%, 20.3%, and 4.9%, separately. Compared to GraRep, HOPE, and SDNE, the Micro-F1 of HGCN* is improved by 8.4%, 11.6%, and 7.4%, and in the Macro-F1, the results are improved by 11.4%, 15.2%, and 9.2%, separately.

To further illustrate the interesting trends, we plot the Micro-F1 and Macro-F1 to investigate the changes of the different baseline methods in Figure 6.

As can be seen from the results in Figure 6, compared to shallow embedding models, matrix decomposition models, and deep embedding models, the Micro-F1 and Macro-F1 of HGCN* are all improved on the Wikipedia dataset. Figure 6 also shows that the Micro-F1 and Macro-F1 of HGCN have been significantly improved by using the max-pooling. The experimental results further indicate that the improved

model for the dataset with a large number of label types can get better graph embedding effectively.

5. Conclusion

In this study, we explore an unsupervised graph embedding method for the undirected graph. Comparing with conventional graph embedding models, we introduce hierarchical graph convolution network to propose the HGCN and HGCN* methods respectively. Besides, we come to the following conclusions:

(1) In this paper, a hierarchical GCN for the undirected graph with only structure information is proposed. The model does not need the extra features and labeled data, which improves the applicability of the GCN in the real network.

(2) Moreover, we introduce max-pooling to improve the proposed model according to the number of label types. Experimental results show that our model achieves considerable improvement than all the baselines.

However, due to the limitations of the pre-training and the clustering methods, the proposed model in this paper still has the disadvantage of obtaining the initial node embedding and the poor label accuracy. In the future, we will further focus on dealing with the problems and plan to explore the impact of initialing models and clustering methods on the proposed model, which ultimately can provide a better method of unsupervised graph embedding.

Data Availability

All experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Thanks are due to the teachers and classmates of the project team for their guidance and help. This research was supported by the National Natural Science Fund of China (no. 61936012 and 61806117), the National Social Science Fund of China (no. 18BYY074), the Scientific and Technological Innovation Programs of Higher Education Institutions in Shanxi (No. 201802012), and the Innovation Project for College Graduates of Shanxi Province (no. 2020SY015).

References

[1] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017, https://arxiv.org/abs/1710.09282.

[2] G. Palash and F. Emilio, "Graph embedding techniques, applications, and performance: a survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[3] K. S. Mehran, G. Rishab, J. Kshitij et al., "Relational representation learning for dynamic (knowledge) graphs: a survey," 2019, https://arxiv.org/abs/1905.11485.

[4] I. K. Fodor, *A Survey of Dimension Reduction Techniques*, Neoplasia, Amsterdam, Netherlands, 2002.

[5] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1–3, pp. 37–52, 1987.

[6] J. B. Kruskal, M. Wish, and E. M. Uslaner, *Multidimensional Scaling, Methods*, Vol. 116, SAGE, New York, NY, USA, 1978.

[7] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in Neural Information Processing Systems*, vol. 14, no. 6, pp. 585–591, 2002.

[8] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: methods and applications," 2017, https://arxiv.org/abs/1709.05584.

[9] C. C. Tu, W. C. Zhang, Z. Y. Liu, and S. M. Sun, "Max-margin deepwalk: discriminative learning of network representation," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3889–3895, New York, NY, USA, July 2016.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[11] C. Yang, Z. Y. Liu, D. L. Zhao, M. S. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2111–2117, Buenos Aires, Argentina, July-August 2015.

[12] X. F. Sun, J. Guo, X. Ding, and T. Liu, "A general framework for content-enhanced network representation learning," 2016, https://arxiv.org/abs/1610.02906.

[13] D. K. Zhang, J. Yin, X. Q. Zhu, and C. Q. Zhang, "Homophily, structure, and content augmented network representation learning," in *Proceedings of the International Conference on Data Mining (ICDM)*, pp. 609–618, Barcelona, Spain, December 2016.

[14] P. Velickovic, W. Fedus, W. L. Hamilton, P. Lio, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[15] Z. N. Hu, C. J. Fan, T. Chen, K. W. Chang, and Y. Z. Sun, "Pre-training graph neural networks for generic structural feature extraction," 2019, https://arxiv.org/abs/1905.13728.

[16] J. Pennington, R. Sorcher, and C. D. Manning, "Glove: global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October 2014.

[17] J. Zhou, G. Q. Cui, Z. Y. Zhang et al., "Graph neural networks: a review of methods and applications," 2018, https://arxiv.org/abs/1812.08434.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 3111–3119, Lake Tahoe, NV, USA, December 2013.

[19] T. Tang and H. Liu, "Leveraging social media networks for classification," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 447–478, 2011.

[20] B. Perozzi, R. Alrfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining (KDD), pp. 701–710, New York, NY, USA, August 2014.

[21] J. Tang, M. Qu, M. Z. Wang, M. Zhang, J. Yan, and Q. Z. Mei, "Line: large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pp. 1067–1077, Florence, Italy, May 2015.

[22] A. Grover and J. Leskovec, "Node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 855–864, San Francisco, CA, USA, August 2016.

[23] L. F. R. Ribeiro, P. H. P. 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 (KDD)*, pp. 385–394, Halifax, Canada, August 2017.

[24] J. Y. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 478–487, New York, NY, USA, June 2016.

[25] H. Y. Xu, H. T. Liu, W. J. Wang, Y. H. Sun, and P. F. Jiao, "NEFLGC: network embedding based on fusing local (First-Order) and global (Second-Order) network structure with node content," *Pacific-Asia Conference on Knowledge discovery and data mining (PAKDD)*, pp. 260–271, 2018.

[26] D. X. Wang, P. Cui, and W. W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1225–1234, San Francisco, CA, USA, August 2016.

[27] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv.org/abs/1609.02907.

[28] W. L. Hamilton, Z. T. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of Annual Conference Neural Information Processing Systems (NIPS)*, pp. 1024–1034, Long Beach, CA, USA, December 2017.

[29] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, https://arxiv.org/abs/1710.10903.

[30] S. Abuelhaija, A. Kapoor, B. Perozzi, and J. Lee, "N-GCN: multi-scale graph convolution for semi-supervised node classification," 2018, https://arxiv.org/abs/1802.08888.

[31] J. Chen, T. F. Ma, and C. Xiao, "FastGCN: fast learning with graph convolutional networks via importance sampling," 2018, https://arxiv.org/abs/1801.10247.

[32] K. Sun, Z. X. Zhu, and Z. C. Lin, "Multi-stage self-supervised learning for graph convolutional networks," 2019, https://arxiv.org/abs/1902.11038.

[33] M. Qu, Y. Bengio, and J. Tang, "GMNN: graph markov neural networks," 2019, https://arxiv.org/abs/1905.06214.

[34] U. Bodenhofer, A. Kothmeier, and S. Hochreiter, "AP cluster: an R package for affinity propagation clustering," *Bioinformatics*, vol. 27, no. 17, pp. 2463-2464, 2011.

[35] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[36] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[37] S. S. Cao, W. Lu, and Q. K. Xu, "Grarep: learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pp. 891–900, Melbourne Australia, October 2015.

[38] M. D. Ou, P. Cui, J. Pei, Z. W. Zhang, and W. W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1105–1114, San Francisco, CA, USA, August 2016.

[39] Qinghua University, OpenNE, https://github.com/thunlp/OpenNE.