DSI DATA SCIENCE AND INFORMETRICS
VOL.3, NO.4, NOV. 2023

RESEARCH ARTICLE

Weighted graph convolutional networks based on network node degree and efficiency

Fenggao Niu*, Yanan Jiang, Cuiyun Zhang

School of Mathematical Sciences, Shanxi University, Taiyuan, China

ABSTRACT

In the study of graph convolutional networks, the information aggregation of nodes is important for downstream tasks. However, current graph convolutional networks do not differentiate the importance of different neighboring nodes from the perspective of network topology when aggregating messages from neighboring nodes. Therefore, based on network topology, this paper proposes a weighted graph convolutional network based on network node degree and efficiency (W-GCN) model for semi-supervised node classification. To distinguish the importance of nodes, this paper uses the degree and the efficiency of nodes in the network to construct the importance matrix of nodes, rather than the adjacency matrix, which usually is a normalized symmetry Laplacian matrix in graph convolutional network. So that weights of neighbor nodes can be assigned respectively in the process of graph convolution operation. The proposed method is examined through several real benchmark datasets (Cora, CiteSeer and PubMed) in the experimental part. And compared with the graph convolutional network method. The experimental results show that the W-GCN model proposed in this paper is better than the graph convolutional network model in prediction accuracy and achieves better results.

KEYWORDS

Graph convolutional network; Network efficiency; Weighted graph convolutional neural network (W-GCN); Text classification

1 Introduction

In recent years, convolutional neural network (CNN) has developed rapidly and attracted extensive attention due to its powerful modeling ability (Zhou et al., 2017). Compared with traditional methods, the arrival of CNN brings new solutions to image processing (Zhang et al., 2019), natural language processing and other fields, such as machine translation (Hu et al., 2015), image recognition (He et al., 2016) and speech recognition (Hinton et al., 2012).

Traditional CNNs have brought improvements in the field of text and image, but they can only process Euclidean spatial data, including images, text, speech, etc., which have translation invariance (in matrix representation). For example, for image data, an image can be represented as a group of regularly distributed pixels in Euclidean space, while translation invariance means that local structures of the same size can be obtained with any pixel as the center. In this case, CNN model local connections by learning the convolution kernel shared

* Corresponding Author: nfgao@sxu.edu.cn

at each pixel, and then learn meaningful hidden layer representations for images.

How about CNNs for non-Euclidean spatial data, for example, graph data? Graph data can naturally represent real-world data structures such as the World Wide Web, transportation networks, and social networks. They are gaining attention because of their ubiquity. Different from image and text data, the local structure of each node in graph data is different, which makes translation invariance no longer satisfied (Shuman et al., 2013). The lack of translation invariance makes it challenging to define CNN on graph data. However, due to the universal existence of graph data, researchers began to focus on how to construct a deep learning model on graphs. The main difficulty lies in the diversity and non-translational invariance of graph data, which makes it a challenging task to define convolution on graph data (Xu et al., 2020).

Machine learning model was not involved in graph data modeling at the earliest, such as PageRank (Page et al., 1999), HITS (Ceglar & Roddick, 2006) and other commonly used algorithms for webpage sorting. Some research work with the knowledge of graph theory, such as using the eigenvalues and eigenvectors of Laplacian matrix for community analysis or membership clustering (Ng et al., 2001). With the rise of deep learning, researchers began to introduce deep learning models into graph data. The representative research work is Network Embedding (Qi et al., 2018), which learns fixed-length representations for each node through the constraint of node proximity, such as DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), Node2vec (Grover & Leskovec, 2016), etc. In this period, when solving specific application problems, researchers usually model them as two-stage problems. Taking node classification as an example, the first stage learns the expression of uniform length for each node, and the second stage takes node expression as input to train the classification model.

Later, the researchers gradually focus on how to transfer deep learning model to the graph data, then do end-to-end modeling. With the capabilities of CNN modeling the local structure and common node dependencies on the graph, graph CNN (GCNN) become one of the most active and important branches of research. Bruna et al. (2013) based on graph signal processing, transformed graph signal from spatial domain to frequency domain through Fourier transform, and then defined convolution operation in frequency domain, and proposed the CNN on graph for the first time. However, this spectral method has the disadvantage of high space and time complexity. Defferrard et al. (2016) defined Chebyshev polynomials of diagonal matrix of feature vectors as filters, that is, the method of fitting convolution kernel with Chebyshev polynomials was used to reduce computational complexity. Kipf & Welling (2016) proposed a simple and effective layered propagation method by using first-order approximation to simplify the calculation of the network model, and directly operating on graph structure data according to spectral graph convolution. Although this method is classified as spectral method, it has begun to define the weight matrix of nodes from the perspective of space. Inspired by this method, spatial methods were applied, and researchers began to consider using attention mechanism and serialization model to model the weight between nodes in graph. In this period, GCNN almost does not consider the characteristics of the graph itself in the process of constructing convolution operator.

As the convolution operator is gradually perfect, research began to consider various graph features, to pay close attention to how to model graph on high order information, and to get fine design in view of the edge feature of graph or heterogeneous graph, etc. Hamilton et al. (2017) proposed GraphSAGE. The model uses multi-level aggregation to obtain information of neighbor nodes, so it can aggregate information of nodes with large distance around as the iteration continues. However, in GraphSAGE, each neighbor node is treated equally, whereas in real scenarios, different neighbor nodes may play different roles on the core node. Graph attention network (GAT)(Velickovic et al., 2018) aggregates neighbor nodes through self-attention mechanism and realizes adaptive matching of weights of different neighbor nodes, thus improving the accuracy of the model. The weight of attention-based graph neural network (AGNN)(Thekumparampil et al., 2018) is obtained by multiplying the cosine similarity of feature vectors of two connected nodes by an adaptive coefficient β, to achieve different weight allocation of neighbor nodes. Wave characteristic model (Wang et al., 2021) uses node features with wavelet function weight to describe node neighborhood. Edge-featured graph attention network (EGAT)(J. Chen & H. Chen, 2021) extend the use of graph neural networks to those tasks learning on graphs with both node and edge features. GRAPE (automorphic equivalence-aware graph neural network)(Xu et al., 2021) uses learnable automorphic equivalence (AE)-aware aggregators to explicitly differentiate the Ego-AE of each node's neighbors with the aids of various subgraph templates. Typed-edge graphlets degree vector (TyE-GDV) (Jia et al., 2022) embed edge type information in graphlets and generate a vector, it focuses on the rich information of the interaction between nodes, namely edge attributes or edge types.

Since it was proposed, GCNN has attracted a lot of attention from researchers. Its application fields include computer science, artificial intelligence, signal processing and other traditional machine learning fields, as well as interdisciplinary research in physics, biology, chemistry, and social sciences. In the study of graph convolutional networks (GCN), how to reasonably weight the importance of each node's neighbor nodes is a key issue. Although good progress has been made, the importance of neighbor nodes has not been considered from the network topology itself. The influence of a node depends not only on the number of neighbors, but also on its own topology. Liu et al.(2021) distinguished the importance of neighbor nodes by the proportion of the sum of all neighbor degrees in the central node when searching for influence nodes by voting mechanism. Niu et al.(2021) constructed weighted short text network with improved node weight based on word co-occurrence analysis. Important degree of nodes in the network has many characterization methods, and this article is constructing weighted graph convolutional neural network (W-GCN) model, by using node degree and efficiency of nodes in the network to construct the node important degree matrix. The W-GCN can distinguish importance degree of neighbor nodes and makes the network focus on the important feature information of neighbor nodes in the process of learning.

2 Related work

2.1 Graph convolutional network

Due to some of the traditional neural network models such as CNN in non-Euclidean data structure does not have translation invariance, namely cannot adopt the same size of convolution kernels for convolution, they will not be able to handle such non-Euclidean graph data. Thus, the GCN arises at the historic moment, made on the structure of irregular graph convolution becomes possible.

Here is the GCN model improved by Kipf & Welling (2016). It is a neural network that operates directly on a graph and generates embedding vectors of nodes according to the information of their neighbor nodes. Firstly, the Laplace matrix of GCN is calculated according to the relevant information of graph (mainly including node information and structure information). Then the graph is convolved to extract the useful part of the original information. Finally, a fully connected neural network is added to classify and judge nodes. The specific process steps are as follows:

We use $G=\{V, E\}$ to represent an undirected graph without self-loop, where $V=\{v_1, v_2, \ldots, v_N\}$ is the set of all nodes; $E=\{(v_1, v_2), i, j=1, 2, \ldots N\}$ is the set of edges between nodes in the graph; Generally, the connection relation between network nodes can be represented by network adjacency matrix, namely, $A=(a_{ij})_{i,j=1,\ldots,N}$. If there is a connection between node v_i and node v_j, then $a_{ij}=1$; otherwise, $a_{ij}=0$. We set that $a_{ii}=0$, $i=1, \ldots N$, which means that the adjacency matrix A is a symmetric matrix with diagonal elements of 0. $L=D-A$ represents the Laplace matrix on the graph. Here $D=diag\{d_1, \ldots, d_N\}$ represents the network degree matrix, where $d_i=\sum_{j=1}^{N} a_{ij}$ is the degree of node v_i. The normalized Laplace matrix is defined as $L=I_n-D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where $I_n \in R^{n\times n}$ is the identity matrix. In addition, we use X to represent node features on graph G, where $X \in R^{n\times D}$, and $X_i \in R^D$ is the feature of the node $v_{i\cdot}$, n is the number of nodes, and D is the number of features of each node.

The Laplacian operator can be obtained according to the Laplacian matrix, and the convolution operation is carried out on the graph structure data. The hierarchical propagation rules are as follows:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

Where , $\tilde{A} = A + I_N$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $H^{(l)}$ is the eigenvalue of the previous state of node, $W^{(l)}$ is the weight matrix of the CNN at the *l-th* layer, and σ is a nonlinear activation function.

Firstly, the multiplication of the adjacency matrix and $H^{(l)}$ is to aggregate the values of the adjacent nodes. In order not to lose the original information of the node, we often force each node to have a self-loop, namely a connection to itself. That's the same thing as taking the adjacency matrix A plus the identity matrix I_N. Noted as matrix $\tilde{A}$. But the matrix $\tilde{A}$ is not normalized, which will lead to a larger scale of node features after each multiplication, so need to apply Laplace symmetric normalization for $\tilde{A}$, namely $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. By means of hierarchical propagation rules, GCNN brings the sharing feature of local parameter of CNN into the graph structure, making the receptive domain of each node improve more with the increase of propagation layers, so as to obtain more information of neighbor nodes.

The GCN improved by Kipf & Welling (2016) is generally set as two layers, and its graph convolution formula is:

$$Z = f(X, A) = softmax(\hat{A} \cdot Relu(\hat{A}XW^{(0)})W^{(1)})$$

Where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. The weights $W^{(0)} \in R^{D\times F}$ are the weight matrix from the input layer to the hidden layer. Similarly, $W^{(1)} \in R^{D\times F}$ is the weight matrix from the hidden layer to the output layer. In the whole training process, since the normalized adjacency matrix is a constant in the training process, it is only necessary to conduct gradient descent training for the weight matrix W of each layer.

2.2 Spatial graph convolution

On the basis of GCN, spatial methods of graph convolution have begun to develop, which aggregate the information of neighboring nodes in different ways and directly apply convolution to the graph. The GraphSAGE model not only focuses on first-order neighborhood information, but also aggregates further node information, allowing nodes to learn more diverse information. However, in GraphSAGE, the importance of each neighboring node is not distinguished, and in practical applications, different neighboring nodes may play different roles in the central node. AGNN has proven that the most critical part of a GCN is the convolutional layer rather than the nonlinear activation layer. Secondly, it proposes to use the cosine similarity of node features to calculate the weights of two adjacent nodes, thereby achieving different weight assignments for neighboring nodes. GAT utilizes attention mechanism to assign weights to neighboring nodes, thereby learning more information about neighboring nodes. When aggregating messages from neighboring nodes, the key issue is how to allocate the weights of each node's neighbors reasonably. Currently, when performing convolution operations from the perspective of spatial methods, the relationships between nodes have not been explored from the perspective of network topology. Therefore, this article focuses on the node degree and efficiency to construct a node importance matrix, in order to weight the GCN.

3 The proposed W–GCN model

Graph structure can be used to represent some non-Euclidean data in the real world. Observing the graph structure data, it can be found that different nodes in the graph often have different importance, and usually the nodes in the central position play a key role in adjacent nodes. If GCN can better pay attention to the feature information of these important neighbor nodes in the process of information aggregation of nodes, it can have a more comprehensive learning of graph structure, and then improve network performance.

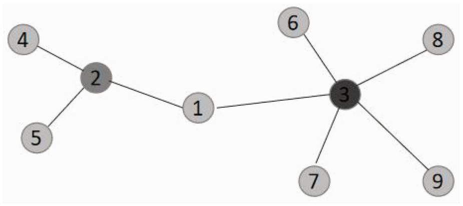
The degree of a node in network refers to the number of nodes directly connected to the node. It is generally believed that the importance of a node has a great relationship with its degree, so the degree of a node can directly reflect its influence on its neighbor nodes. The larger the degree value is, the greater the importance of the node contribution to its neighbor nodes is.

Network efficiency E refers to the average sum of reciprocal distances between all node pairs in the network, which represents the average difficulty of network information flow. The higher the network efficiency is, the easier the network information flow is (Zhou et al., 2016).

$$E = \frac{1}{n(n-1)} \sum_{i \neq j} \frac{1}{d_{ij}}$$

Where n is the number of nodes in the network, and d_{ij} is the distance between nodes v_i and v_j . The distance between nodes refers to the number of edges on the shortest path between two nodes. If there is no path between v_i and v_i , then $d_{ij} \to \infty$.

In this paper, the nodes in graph structure data are weighted according to their degree and efficiency in network. Figure 1 is an example.



Note: The depth of the color indicates the weight of the node

Figure 1 The central node aggregates information about neighbor nodes based on their weights

As shown in Figure 1, we use the depth of the color to indicate the weight. As to the node 1, which have two neighbor nodes, 2 and 3. There are four neighbor nodes for node 3, which has higher efficiency than other nodes (for example node 2). In the processing of aggregation information for node 1, if more information about node 3 is taken, it means more information in the graph is considered. Therefore, this paper constructs node importance degree matrix, combining the node degrees and its efficiency, which considers its role in the network information flow and its contribution to the adjacent nodes. And then normalize the matrix through softmax function for convolution operation, instead of Laplace normalization of adjacency matrix. It enables nodes to well aggregate information about different neighbor nodes.

The construction process of node importance matrix is as follows:

Step 1: Importance contribution matrix (Liu et al., 2021). In an undirected network with *n* nodes, if there is a link between node v_i and v_j , but the nodes have different importance to each other, when considering the importance of neighbor nodes to the central node, we first calculate the sum of the degrees of all neighbor nodes of the central node, and then distinguish the importance of neighbor nodes with the proportion of each neighbor node in the sum of all neighbor nodes. If the degree of node v_i is D_i, the importance of node v_i to v_j node is expressed by $D_i / \sum_{v_k \in N(v_j)} D_k$. $N(v_j)$ is the neighbor node set of nodes v_j, and v_k is the k-th neighbor node in the neighbor node set of nodes v_j . The importance contribution ratio values of all nodes to their neighboring nodes are shown in a matrix, thus forming the importance contribution matrix of available nodes, denoted as H_{IC} :

$$H_{IC} = \begin{bmatrix} 1 & \frac{\delta_{12} D_2}{\sum_{v_j \in N(v_1)} D_j} & \cdots & \frac{\delta_{1n} D_n}{\sum_{v_j \in N(v_1)} D_j} \\ \frac{\delta_{21} D_1}{\sum_{v_j \in N(v_2)} D_j} & 1 & \cdots & \frac{\delta_{2n} D_n}{\sum_{v_j \in N(v_2)} D_j} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\delta_{n1} D_1}{\sum_{v_j \in N(v_n)} D_j} & \frac{\delta_{n2} D_2}{\sum_{v_j \in N(v_n)} D_j} & \cdots & 1 \end{bmatrix}$$

The element on the diagonal of the matrix is *l*, indicating that the contribution ratio of the node to its own importance is 1. Node importance contribution matrix H_{IC} has the same structure as adjacency matrix and is regarded as a mapping of it. The mapping rule is as follows:

$$\begin{cases} \delta_{ij} \to \delta_{ij} D_j / \sum_{v_l \in N(v_i)} D_l, i \neq j \\ \delta_{ij} \to 1, i = j \end{cases}$$

Where δ_{ij} is the contribution allocation parameter. If v_i and v_j are directly connected, the value is 1 ; otherwise, the value is 0.

Step 2: Node importance matrix H_E. The degree is used to construct the importance correlation between nodes, and the efficiency of nodes is used to represent the location information of nodes.

The efficiency of node *K* is I_k , where $I_k = \frac{1}{n} \sum_{i=1, i \neq k}^{n} \frac{1}{d_{ki}}$. Node efficiency represents the difficulty in information transmission of a node to other nodes, and using node efficiency to measure the global importance of a node is a reasonable indicator. Therefore, the larger the efficiency value of node is, the more important the node's position is in the network information transmission process. Thus, the efficiency of nodes reflects the importance of nodes to a certain extent.

The combination of the efficiency values and the importance contribution value is used to replace the importance contribution ratio value of nodes in H_{IC}, and then get the importance evaluation matrix of nodes.

$$H_E = \begin{bmatrix} I_1 & \frac{\delta_{12} D_2 I_2}{\sum_{v_j \in N(v_1)} D_j} & \cdots & \frac{\delta_{1n} D_n I_n}{\sum_{v_j \in N(v_1)} D_j} \\ \frac{\delta_{21} D_1 I_1}{\sum_{v_j \in N(v_2)} D_j} & I_2 & \cdots & \frac{\delta_{2n} D_n I_n}{\sum_{v_j \in N(v_2)} D_j} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\delta_{n1} D_1 I_1}{\sum_{v_j \in N(v_n)} D_j} & \frac{\delta_{n2} D_2 I_2}{\sum_{v_j \in N(v_n)} D_j} & \cdots & I_n \end{bmatrix}$$

Where element of $H_E(i, j)$ represents the importance contribution value of node v_j to node v_i. It can be seen that the importance contribution value of a node to its neighbor nodes is related to its own efficiency and degree value. The greater the efficiency value and degree value of a node are, the greater the importance contribution value of its neighbor nodes is.

In addition, adding an adaptive coefficient β (Thekumparampil et al., 2018), with one β for each aggregation layer, makes it possible to impose a higher weight to the relevant important neighbors.

The constructed matrix P and adjacency matrix A have the same form, P_{ij} is for the two nodes without edge connection, and $P_{ij} = \beta * H_{Eij}$ for the two nodes with edge connection.

In this paper, the settings of convolutional layers are consistent with those of GCNs. The convolution formula is:

$$Z = f(X, P) = softmax\big(P \cdot Relu\big(PXW^{(0)}\big)W^{(1)}\big).$$

4 Experiments

4.1 Benchmark datasets

The proposed W-GCN model is examined on semi-supervised classification tasks using three reference datasets of graph neural network, namely Cora, CiteSeer and PubMed of citation network. The citation network dataset consists of text as nodes and citation links as directed edges. Each node has a manual annotation topic from a finite set and has a feature vector. Although the network is directed, we used an undirected version of the graph in all our experiments, which is common in baseline approaches. Statistical descriptions of datasets are summarized in Table 1.

Table 1 The properties of various graph datasets used for the semi-supervised classification task

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
CiteSeer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
PubMed	Citation network	19,717	44,338	3	500	0.003

4.2 Experimental settings

In this part, we only train and test the W-GCN model, a new GCN based on node importance. Instead of using validation set labels in training, we use them to optimize hyperparameters such as dropout rate, learning rate, and L2-regularization factors. According to the previous method of Kipf & Welling (2016), we divide graph data into training set, verification set and test set. In this model, the graph convolution layer and GCN settings are the same, with one convolution layer and one SoftMax layer. The learning rate was set to 0.005, the number of hidden layers was set to 32, and the dropout rate was set to 0.5.

4.3 Experimental results and analysis

In the following experiments, standard benchmark data splits are used just as those have done in the relative literature. All experiments were run on the same fixed partition, which with 20 labeled nodes per class, 500 for validation, 1000 for testing, and the rest as unlabeled data.

In text classification, the evaluation index can be used to judge the performance of the classifier and to analyze the model. The common indexes of classification and evaluation are accuracy, recall and precision. True positive (*TP*) means to predict a positive class as a positive class. False negative (*FN*) means to predict a positive class as a negative class. False positive (*FP*) means to predict a negative class as a positive class. True negative (*TN*) means to predict a negative class as a negative class. Accuracy is a statistic for all samples expressed as:

$$Accuracy = \frac{TP + FN}{TP + TN + FP + FN} \times 100\%$$

The accuracy results of each method on three datasets are shown in Table 2. The accuracy of the baseline methods compared in the experiments are obtained from the existing literature. If baseline results are not reported in relative literature, we leave them blank in the table rather than running these experiments on untuned parameters ourselves. The visualization of classification in CiteSeer and Cora datasets after dimensionality reduction are show in Figure 2 and Figure 3.

Table 2 Performance comparisons of semi-supervised classification methods (%)

Methods	Cora	CiteSeer	PubMed
T–SVM	57.5	64.0	62.2
DeepWalk	67.2	43.2	65.3
Node2vec	74.9	54.7	75.3
LP	68.0	45.3	63.0
DCNN	76.8	—	73.0
GCN	81.5	70.3	79.0
AGNN	83.1	71.7	79.9
GAT	83.0	72.5	79.0
W–GCN (in this study)	83.1	72.2	79.0

Note: "—": not reported in relative literature. T-SVM: tr ansductive support vector machines; LP: label propagation; DCNN: dynamic convolution neural network; GCN: graph convolutional network; AGCN: adaptive graph convolutional neural network; GAT: graph attention network; W-GCN: weighted graph convolutoin neural network.

Table 2 showed that it is very important to make use of the structure and node features of graphs simultaneously in semi-supervised learning of graphs. Methods that use both the structure and node characteristics of the graph get more accurate results than those do not. For example, T-SVM as a semi-supervised method, use only labeled and unlabeled tags and node characteristic attributes. DeepWalk, based on Skip-Gram, ignore node characteristic attributes, and only use tag and graph structure information. So, their accuracy is low.

The W-GCN, based on network topology to assign different weights to nodes respectively, surpass the GCN. It shows a 1.6% improvement in accuracy in the Cora dataset and 1.9% improvement in the CiteSeer dataset. The accuracy in PubMed dataset is the same as GCN. This shows that the method proposed in this paper can better allocate the weights of nodes to some extent, because it pays more attention to the feature information of important neighbor nodes than others in the process of network learning.

Combining Table 1 and Table 2, the number of nodes may be an influencing factor for the accuracy. The W-GCN get significant improvement over GCN in Cora and CiteSeer datasets, which with relatively fewer nodes. Meanwhile, the accuracy of W-GCN in CiteSeer data sets is also slightly higher than that of AGNN. These indicate that the proposed method has certain advantages. But the accuracy of W-GCN and GCN are the same in PubMed dataset, which with more nodes in the graph. These show that, the more nodes in graph, the lower the efficiency of each node, and the less the recognition ability of the importance of neighbor nodes. Therefore, in the follow-up work, we consider using local efficiency to construct the node importance degree matrix of nodes. By this way, the importance of neighbor nodes may be better distinguished and classification accuracy may be improved.

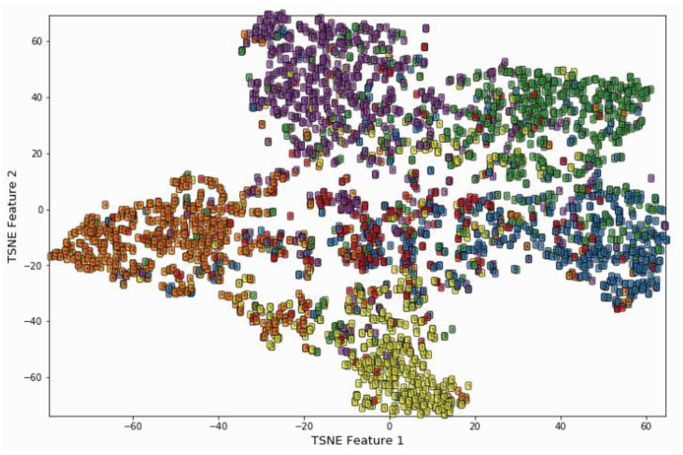


Figure 2 Visualization of nodes classification in CiteSeer data after dimension reduction

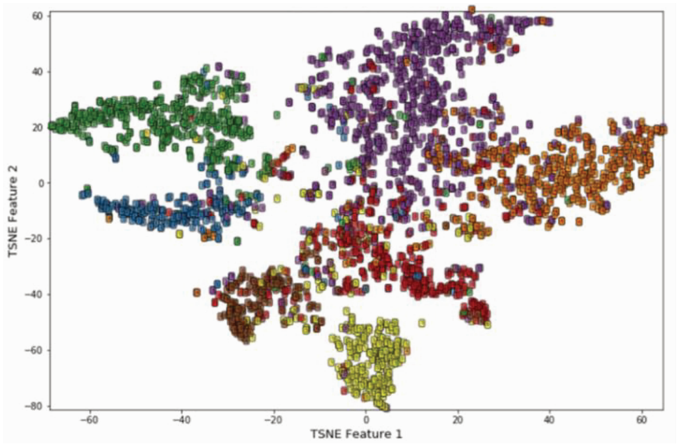


Figure 3 Visualization of nodes classification in Cora data after dimension reduction

5 Conclusions

In the graph node classification task, the convolution operator of graph convolution is the key part, which is related to whether the whole network can learn information of each neighbor nodes more reasonably. That is also the focus of research in the field of GCN and its variants.

In this paper, we propose a new method, which assigns different weights to each neighbor node to complete the semi-supervised node classification task on the graph. The W-GCN focus on the importance of the node in the network and the contribution to the central node, and able to distinguish whether a node is important or not. The important node gains more weight than others, which make the convolution operator get more important information of neighbor nodes. The superiority and effectiveness of W-GCN are also proved over GCN and other models through the experiment on three benchmark datasets. In addition, the performance of the model is related to the size of the graph.

In the further research, it is considered that W-GCN should be applied to the task of text classification, and it would get better effect on the problems. At the same time, better method should be studied to distinguish the importance of nodes. From several parts, such as the feature extraction of each layer of nodes, feature fusion, better use of node information, to ensue improve the accuracy of the node classification.

Acknowledgements

This work was mainly supported by Fundamental Research Program of Shanxi Province (No. 202203021211305) and Shanxi Scholarship Council of China (2023-013).

References

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *ArXiv*, 1312.6203. https://doi.org/10.48550/arXiv.1312.6203

Ceglar, A., & Roddick, J. F. (2006). Association Mining. *ACM Computing Surveys, 38* (2), 2.1–2.42.

Chen, J., & Chen, H. (2021). Edge –featured graph attention network. *ArXiv*, 2101.07671. https://doi.org/10.48550/arXiv.2101.07671

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems, 29*, 3844–3852. https://doi.org/10.48550/arXiv.1606.09375

Grover, A., &Leskovec, J. (2016). node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–864. https://doi.org/10.1145/2939672.2939754

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Proceedings of the 31st International Conference on Neural Information Processing Systems,* 1025–1035. https://doi.org/10.48550/arXiv.1706.02216

He, K., Zhang, X. Y., Ren, S. Q., & Su, J. (2016). Deep residual learning for image recognition. 2016 *IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Hinton, G.E., Deng, L., Yu, D., Dahl, G.E., Mohamed, A., Jaitly, N., Senior, A.W., Vanhoucke, V., Nguyen, P., Sainath, T.N., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine, 29* (6), 82–97. https://doi.org/10.1109/MSP.2012.2205597

Hu, B., Lu, Z., Hang, L. & Chen, Q. (2015). Convolutional neural network architectures for matching natural language sentences. *Advances in Neural Information Processing Systems*, 3. https://doi.org/10.48550/arXiv.1503.03244

Jia, M., Van Alboom, M., Goubert, L., Bracke, P., Gabrys, B., & Musial, K. (2022). Encoding edge type information in graphlets. *PLOS ONE*. https://doi.org/10.1371/journal.pone.0273609

Kipf, T. N., & Welling, M. (2016). Semi–supervised classification with graph convolutional networks. *ArXiv,* 1609.02907. https://doi.org/10.48550/arXiv.1609.02907

Liu, P., Li, L., Fang, S., & Yao, Y. K. (2021). Identifying influential nodes in social networks: A voting approach. *Chaos Solitons & Fractals, 152* (7415), 111309. https://doi.org/10.1016/j.chaos.2021.111309

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: analysis and an algorithm. *In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic* (NIPS´01), 849 – 856.

Niu. F. G., & Gao, X. X. (2021). An improved Chinese short text similarity measurement model based on weighted network. *Journal of Information Science, 40* (3), 278–285.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. *Stanford Digital Libraries Working Paper*.

Perozzi, B., Al–Rfou, R., & Skiena, S. (2014). DeepWalk: Online learning of social representations. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710. https://doi.org/10.1145/2623330.2623732

Qi, J. S., Liang, X., Li, Z. Y., Chen, Y. F., & Xu, Y. (2018). Representation learning of large–scale complex information network: Concepts, methods and challenges. *Chinese Journal of Computer, 41* (10), 2394–2420. https://doi.org/10.11897/SP.J.1016.2018.02394

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high–dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine, 30* (3), 83–98. https://doi.org/10.1109/MSP.2012.2235192

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large–scale information network embedding. *Proceedings of the 24th International Conference on World Wide Web*, 1067–1077. https://doi.org/10.1145/2736277.2741093

Thekumparampil, K. K., Wang, C., Oh, S., & Li, L. J. (2018). Attention–based graph neural network for semi–supervised learning. *ArXiv*, 1803.03735. https://doi.org/10.48550/arXiv.1803.03735

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Pietro Liò, P., & Bengio, Y. (2018). Graph attention networks. *ArXiv*,1710.10903. https://doi.org/10.48550/arXiv.1710.10903

Wang, L. L., Huang, C.H., Ma, W. C., Cao, X. Y., & Vosoughi, S. (2021). Graph embedding via diffusion–wavelets–based node feature distribution characterization. *ArXiv*, 2109.07016. https://doi.org/10.48550/arXiv.2109.07016

Xu, B. B., Cen, K. T., Huang, J. J., Shen, H. W., & Cheng, X. Q. (2020). A survey on graph convolutional neural network. *Chinese Journal of Computers, 43* (5), 755–780. https://doi.org/10.11897/SP.J.1016.2020. 00755

Xu, F. L., Yao, Q. M., Hui, P., & Li, Y. (2021). Automorphic equivalence–aware graph neural network. *arXiv*, 2011.04218. https://doi.org/10.48550/arXiv.2011.04218

Zhang, S., Gong, Y. H., & Wang, J. J. (2019). The development of deep convolution neural network and its applications on computer vision. *Chinese Journal of Computers, 42* (3), 453–482. https://doi.org/10.11897/SP.J.1016.2019.00453

Zhou, F. Y., Jin, L. P., & Dong, J. (2017). Review of convolutional neural network. *Chinese Journal of Computers, 40* (6), 1229–1251. https://doi.org/10.11897/SP.J.1016.2017.01229

Zhou, X., Zhang, F. M., Li, K. W., Hui, X. B., Wu, H. S. (2012). Finding vital node by node importance evaluation matrix in complex networks[J]. *Acta Physica Sinica, 61* (5), 050201. https://doi.org/10.7498/aps.61.050201