IET Image Processing

Special Section: Adversarial Learning in Image Processing

IET Journals
The Institution of Engineering and Technology

Two-order graph convolutional networks for semi-supervised classification

ISSN 1751-9659
Received on 21st September 2018
Revised 15th December 2018
Accepted on 11th March 2019
E-First on 6th August 2019
doi: 10.1049/iet-ipr.2018.6224
www.ietdl.org

Fu Sichao[1], Liu Weifeng[1] ✉, Li Shuying[2], Zhou Yicong[3]

[1]*College of Information and Control Engineering, China University of Petroleum (East China), Qingdao 266580, People's Republic of China*
[2]*School of Automation, Xi'an University of Posts & Telecommunications, Xi'an 710121, People's Republic of China*
[3]*Department of Computer and Information Science, University of Macau, Macau 999078, People's Republic of China*
✉ *E-mail: liuwf@upc.edu.cna*

Abstract: Currently, deep learning (DL) algorithms have achieved great success in many applications including computer vision and natural language processing. Many different kinds of DL models have been reported, such as DeepWalk, LINE, diffusionconvolutional neural networks, graph convolutional networks (GCN), and so on. The GCN algorithm is a variant of convolutional neural network and achieves significant superiority by using a one-order localised spectral graph filter. However, only a one-order polynomial in the Laplacian of GCN has been approximated and implemented, which ignores undirect neighbour structure information. The lack of rich structure information reduces the performance of the neural networks in the graph structure data. In this study, the authors deduce and simplify the formula of two-order spectral graph convolutions to preserve rich local information. Furthermore, they build a layerwise GCN based on this two-order approximation, i.e. two-order GCN (TGCN) for semi-supervised classification. With the two-order polynomial in the Laplacian, the proposed TGCN model can assimilate abundant localised structure information of graph data and then boosts the classification significantly. To evaluate the proposed solution, extensive experiments are conducted on several popular datasets including the Citeseer, Cora, and PubMed dataset. Experimental results demonstrate that the proposed TGCN outperforms the state-of-art methods.

1 Introduction

With the rapid development and popularisation of computer and information technology, the big data is gradually receiving the attention and has become a hot research topic in all walks of life. With the in-depth study of big data, the application field of big data is constantly expanding [1–4], such as agriculture, business and medical treatment. As a typical technique for feature learning of big data, deep learning uses end-to-end learning to extract data information, which has been successfully applied to speech recognition [5, 6], image processing [7, 8], and natural language processing [9, 10].

There are many kinds of deep learning algorithms that have been developed in recent years including DeepWalk, LINE, diffusion-convolutional neural networks (DCNNs), and graph convolutional network (GCN), which are applied to the structure information embedding of data. Perozzi *et al.* [11] proposed a new method to learn the latent relationships of data through the SkipGram model, i.e. DeepWalk. This method selects network nodes randomly and uniformly through random walk, which can generate a fixed-length random walk sequence. However, it does not present a clear objective function. Random walk needs a lot of time and leads to high computation cost for the model.

Tang *et al.* [12] proposed a model to assimilate the local and global structure information, which was named LINE. The model maps the nodes in a large network to the vector space according to the density of one-order and two-order relationships for nodes, so that the closely connected nodes are projected into similar locations. However, it is difficult to obtain highly nonlinear structure information of data for the shallow model LINE.

Atwood and Towsley [13] proposed new DCNNs for node classification through the diffusion-convolutional operation. This model offers us a flexible representation of node data that encodes node structure information. However, it cannot be readily applied to large node data. It fails to express the useful spatial dependency relationships between nodes.

Kipf and Welling [14] proposed a deep learning method of the graph structure data for semi-supervised classification, which is named GCN. This model is based on a one-order polynomial of the spectral convolutions on graph structure data. The structure information of direct neighbour is embedded into its own feature information. Finally, the information of each sample becomes very rich. Unlike other models, GCN does not directly use a k-order polynomial filter and obtain a k-order polynomial filter by accumulating a one-order polynomial of the Laplacian. When GCN model performs a convolution operation on the central nodes, it only uses the nearest-neighbour nodes that are one step away from the corresponding central nodes. After repeating the same convolution operation on the first layer convolution result, it can be extended to the second layer neighbour nodes and so on. However, this model only considers the structure information of the indirect neighbours. It cannot fully consider the manifold structure of the graph structure data and causes the lack of data information.

In order to solve above problems and learn plenty of localised structure information, in this paper, we propose a novel method, i.e. two-order GCN (TGCN), which is based on a two-order polynomial of the Laplacian. In particular, we simplify the definition of a two-order spectral graph convolution and get an extensible formula. We can get the structure information not only from the direct neighbours but also from the undirect neighbours through this model. In each convolution layer, we combine the structure information with the feature information. Finally, TGCN model can extract more representative sample features to represent graph data. In order to prove the validity of the TGCN model, we conduct a large number of experiments on the Citeseer, Cora, and PubMed dataset. A lot of experimental results proves that the TGCN model outperforms the related deep learning model, including GCN and Planetoid.

The remainder of this paper is organised as follows: Section 2 briefly describes the related works of the TGCN model. Section 3 presents the proposed TGCN algorithm in detail. Experimental parameters and experimental results are shown in Section 4. Section 5 finally concludes this paper.

2 Related work

Graph structure data [15–18] has been a hot research topic because of its massive existence in real life. Many classification methods have achieved promising performance on the graph structure data, such as social network datasets, chemical compound datasets, and protein datasets. Our model TGCN not only gets the hints from the graph theory but also from the neural network on the graph structure data domains, which is based on the spectral graph convolution. In this section, we briefly describe several related works of the TGCN in turn.

2.1 Graph theory

In general, graph theory is the theory of studying graph structure data. A graph is a mathematical model that describes the relationships between things and has three essential elements: Nodes, edges, and the weight of the edges. We take an interest in processing signals that are defined on an undirected, connected, weighted graph $G = \{V, E, \boldsymbol{W}\}$, where V is a finite set of vertices with $V = \{v_1, v_2, v_3, \ldots v_n\}$, E is a set of edges with $E = \{e_1, e_2, e_3, \ldots e_n\}$, and $\boldsymbol{W}$ is a weighted adjacency matrix with $\boldsymbol{W} = \{w_1, w_2, w_3, \ldots w_n\}$. The vertices of the graph can be any actual or abstract individual, while the edges are an abstract description of the relationships between the two vertices.

Given a set of vertices V, there are three commonly used methods for constructing a graph: (i) *k nearest-neighbour graph*: Each vertex in the graph is only connected to its k nearest neighbours. (ii) *Full graph*: Each node in the graph connects with all other nodes. (iii) *Hyper-graph*: The essential feature of a hyper-graph is its hyper-edge, which can connect more than two vertices (including two).

Suppose $\{(x)_i\}_{i=1}^{n}$ denotes the vertices. Let M expresses a set of vertices, which have the same class and N denotes a series of vertices that belong to different classes, i.e.

$$M = \{(x_i, x_j) | x_i \text{ and } x_j \text{ have the same labels}\}, \tag{1}$$

$$N = \{(x_i, x_j) \big| x_i \text{ and } x_j \text{ have the different labels}\} \tag{2}$$

Therefore, it also has two types of weighted adjacency matrices: (i) *Zero–one weighting method*: The distance between the nodes that are not connected is zero and the distance between the connected nodes is one. (ii) *Gaussian weighting method*: If the node x_i is connected to x_j, the distance is a specific value, else the distance is zero, where dist(i, j) is the distance between the two nodes:

$$W1_{ij} = \begin{cases} 1 & \text{if } (x_i, x_j) \in M \\ 0 & \text{if } (x_i, x_j) \in N \end{cases} \tag{3}$$

$$W2_{ij} = \begin{cases} \exp\left(-\dfrac{[\text{dist}\,(i,j)]^2}{2\theta^2}\right) & \text{if dist}\,(i,j) < k \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

An essential operator in graph signal processing (GSP) is the non-normalised graph Laplacian, i.e. defined as $L = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D}$ is the degree matrix with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{W}_{ij}$.

We also use the normalised graph Laplacian with $L = \boldsymbol{I}_N - \boldsymbol{D}^{-(1/2)}\boldsymbol{W}\boldsymbol{D}^{-(1/2)}$, where $\boldsymbol{I}_N$ is the identity matrix. Due to the non-normalised and normalised graph Laplacians being the positive semi-definite matrices, it has a complete set of orthonormal eigenvectors $\{(u)_i\}_{i=0}^{n-1}$ and non-negative eigenvalues $\{(\lambda)_i\}_{i=0}^{n-1}$.

Then we can get the eigendecomposition of $L = \boldsymbol{U} \wedge \boldsymbol{U}^{\mathrm{T}}$ through the eigenvectors u_{i} and eigenvalues λ_i, where $\boldsymbol{U} = [u_1, u_2, u_3, \cdots u_n]$ are the graph Fourier bases and $\wedge = [\wedge_1, \wedge_2, \wedge_3, \ldots \wedge_{\mathrm{n}}]$ are the graph frequencies.

The emerging research field of GSP combines graph theory with harmonic analysis. A goal is to build a bridge between signal processing and graph theory [19–21]. It aims to sum up the elementary analysis operations for processing data from Euclidean domains to graph structure data. Under this circumstance, Hammond *et al.* [22] proposed the construction of wavelet transform functions from the nodes of an arbitrary finite weighted graph. It is based on the spectral eigendecomposition of the graph Laplacian L.

2.2 Neural network on graph structure data domains

The model of the spatial convolution method selects a fixed-size neighbour relationship for convolution. In fact, the samples in the graph structure data usually have different number of neighbours. It is an enormous challenge for spatial convolution method. To solve this problem, the concept of spectral convolution is proposed in [23], which is defined as the multiplication of a signal X (a vector of each node) with a filter g_θ in the Fourier domain, which is given as follows:

$$g_\theta(L) * \boldsymbol{X} = \boldsymbol{U} g_\theta(\wedge)\boldsymbol{U}^{\mathrm{T}}\boldsymbol{X} \tag{5}$$

where $\boldsymbol{U}$ is the matrix of the orthonormal eigenvectors of non-normalised graph Laplacian. $\wedge$ is the matrix of the diagonal eigenvalues of L. We can treat $g_\theta(\wedge)$ as the function of eigenvalue $\wedge$.

The above spectral filtering of graph signals (5) have some limitations: (i) Their computational complexity is $O(N^2)$, hence there is an increase in time with the increase of data and (ii) it is non-localised in the vertex domain, i.e. the non-localised-based spectral filter uses all neighbour relationships of each sample in the graph structure data.

This problem can be solved through the polynomial parameterisation of localised filters. So it can get a new representation function of the filter and is denoted as

$$g_\theta(\wedge) = \sum_{k=0}^{k} \theta_k \wedge^k \tag{6}$$

Spectral filter can be represented as the k-order polynomial in the Laplacian. The learning complexity of filter becomes $O(K)$ from $O(N)$. The above method solves the second limitation, but the eigendecomposition of the graph Laplacian L is still very complicated for the data of large graphs.

To solve this problem, Hammond *et al.* [22] proposed a new polynomial, i.e. Chebyshev polynomial. The parameter $g_\theta(\wedge)$ may be computed through the recursive relationship

$$g_\theta(\wedge) = \sum_{k=0}^{k} \theta_k T_k(\tilde{\wedge}) \tag{7}$$

Here, $\tilde{\wedge} = (2/(\lambda_{\max})) \wedge - I_N$, where $\lambda_{\max}$ is the maximum eigenvalue of graph Laplacian L and θ is the Chebyshev coefficient. The definition of Chebyshev polynomial is $T_0(x) = 1, T_1(x) = x$ and $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$. The computational complexity of this method is $O(K|\varepsilon|)$ and is similar to classical ConvNets.

The optimised spectral filter uses the form of K-order polynomial to express the neighbour information that K steps away from a sample. Now, it can get a new definition of spectral convolution, we now have

$$g_\theta(L) * \boldsymbol{X} = \sum_{k=0}^{k} \theta_k T_k(\tilde{L})\boldsymbol{X} \tag{8}$$

Here, $\tilde{L} = (2/\lambda_{\max})L - \boldsymbol{I}_N$, $T_0(\tilde{L}) = 1, T_1(\tilde{L}) = \tilde{L}$, and $T_k(\tilde{L}) = 2\tilde{L}T_{k-1}(\tilde{L}) - T_{k-2}(\tilde{L})$. It defines a new convolution network model on graph structure data through this method, which provides the necessary mathematical background and efficient computational foundations. It can be an effective method to extract local features through novel convolutional layers. The

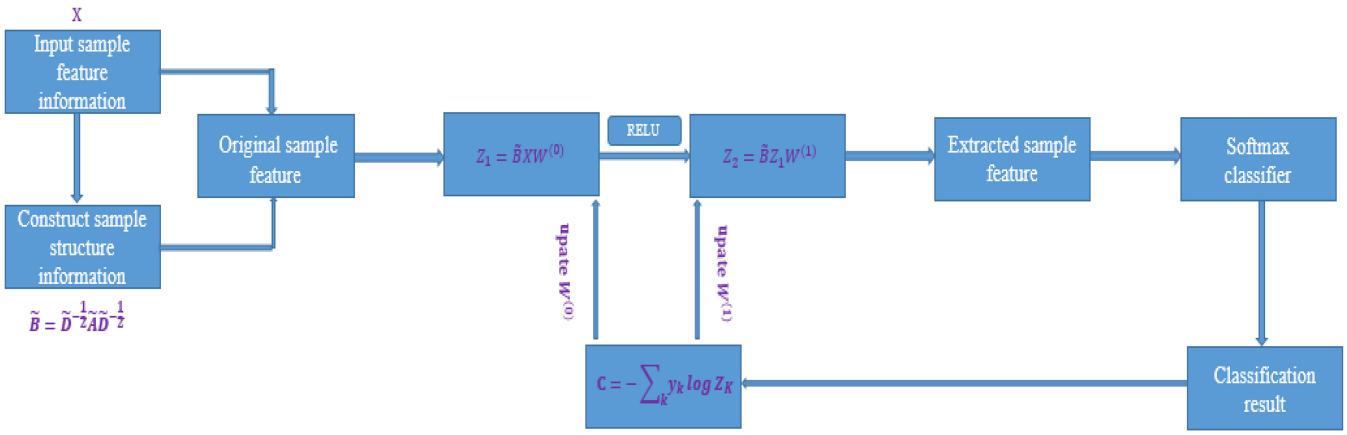

Fig. 1 *Framework of a two-layer GCN model*

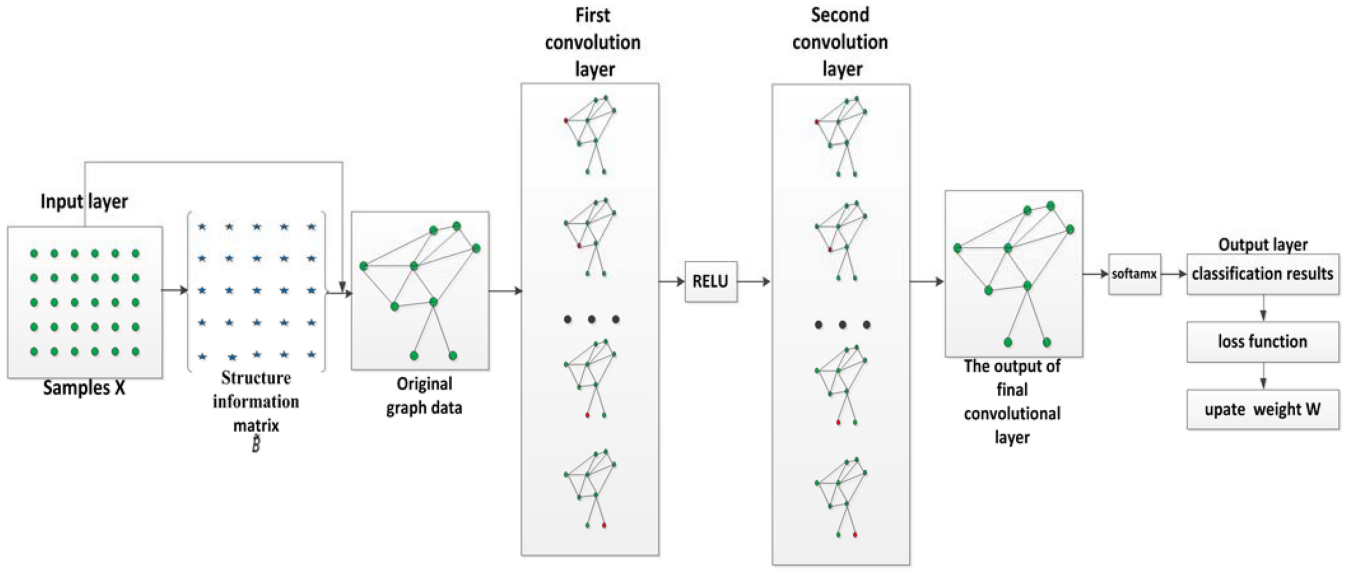

Fig. 2 *Framework of a two-layer TGCN model*

computational complexity of this model is linear and it has the same learning complexity as with convolutional neural networks (CNNs), which is applicable for any graph structure data.

Kipf and Welling[14] proposed a new model through a one-order approximation of spectral filters on graph structure data, which is named as GCN. GCN is a natural extension of the convolutional neural network on graph domain. It can achieve end-to-end learning of the node feature information and structure information at the same time and is currently the best choice for graph structure data learning tasks. It applies to nodes and graphs of any topology structure.

The GCN model can be built through multi-layers of spectral graph convolution. It introduces a layerwise rule, i.e. it limits the number of polynomial orders ($K = 1$). The optimisation process of the layerwise rule can be found in [14]. Fig. 1 shows the framework of a two-layer GCN model:

$$\boldsymbol{H}^{(L+1)} = \sigma\left(\tilde{\boldsymbol{D}}^{-(1/2)}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-(1/2)}\boldsymbol{H}^{(L)}\boldsymbol{W}^{(L)}\right) \tag{9}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$ is the matrix of the structure information on graph structure data. $\boldsymbol{A}$ is the adjacency matrix of sample relationships on graph structure data, but it does not include the relationships of self-node. There are many ways to construct adjacency matrix $\boldsymbol{A}$. $\tilde{\boldsymbol{D}}$ is the degree matrix with $\boldsymbol{D}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$. $\boldsymbol{W}^{(L)}$ is the weight matrix of the specific layer. $\boldsymbol{H}^{(L)}$ is the feature information matrix in some layer with $\boldsymbol{H}^{(0)} = \boldsymbol{X}$. Compared with the basic CNN [24, 25], it can assimilate the structure information and feature information of data at the same time to extract data feature.

3 Two-order graph convolutional networks

Inspired by the definition of spectral graph convolution, we proposed two-order GCN. The TGCN model consists of an input layer, multiple convolution layers, and an output layer. The input layer is the input of the original sample information, which is expressed through various kinds of sample extracted methods. Specifically, in this section, the algorithm part can be roughly divided into two parts. The convolution layer of TGCN is presented first. We get a scalable definition of two-order spectral convolution on graph data and demonstrate how to embed the structure information into the feature information to represent sample features. In each convolution layer, the neighbour nodes are one step or two steps away from the corresponding central node. Thus, each central node contains the sample information of indirect and undirect neighbours. Due to the rich graph data structure information, the sample information that is extracted by the model will be more representative. Then the Softmax layer of TGCN is given. The Softmax layer is a process of multi-class classification for the output features of the last layer. Fig. 2 shows the framework of a two-layer TGCN model. The following will be introduced in turn in this section.

3.1 First convolution layer of TGCN

The GCN model uses a one-order polynomial of Laplacian to assimilate the structure information of each node. It aims to represent graph structure data by mixing the feature information of nodes with structure information. However, each node only gets the feature information of the direct neighbours to enrich self-information. This will result in the data information that is finally

extracted by the neural network, not rich enough, and it finally increases error rate of model.

To learn more localised structure information of graph structure data, we introduced a novel layerwise linear model (two-order polynomial), i.e. we only obtain the structure information that is two steps away from the central node for any nodes. Let us introduce the derivation and simplification of the formula for this model in turn.

In this model, we take $K = 2$ and $\lambda_{\max} \simeq 2$, so that we can get the simplified formula of (8). $\lambda_{\max}$ can be obtained by computing the eigenvalue of the normalised graph Laplacian and is universal. It is named TGCN-1 and is denoted as

$$g_\theta(L) * \boldsymbol{X} = \theta_0 \boldsymbol{X} + \theta_1 (L - \boldsymbol{I}_N)\boldsymbol{X} + \theta_2 [2(L - \boldsymbol{I}_N)^2 - \boldsymbol{I}_N]\,\boldsymbol{X} \quad (10)$$

It has three parameters θ_0, θ_1 and θ_2. The parameters of filter can be shared over in the model. It can get two-order localised neighbour structure information through the filter of this formula form. However, this filter has three parameters, in practice, the computational complexity of the network is more complicated by using this form of the filter. To prevent overfitting and to reduce computational complexity, we can get the following expression that is named TGCN-2:

$$g_\theta(L) * \boldsymbol{X} = \boldsymbol{\theta}\,[(\boldsymbol{I}_N - L) + 2\,(L - \boldsymbol{I}_N)^2]\,\boldsymbol{X} \quad (11)$$

It has one parameter $\theta = \theta_0 = -\theta_1 = \theta_2$ and the above expression also can be written as

$$g_\theta(L) * \boldsymbol{X} = \boldsymbol{\theta}\,[\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)} + 2\,(\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)})^2]\,\boldsymbol{X} \quad (12)$$

It has one parameter $\theta = \theta_0 = -\theta_1 = \theta_2$. In practice, it results in numerical instabilities and exploding/vanishing gradients in the process of building multi-layers convolution network model by using this expression.

To avoid this problem, we use three optimisation methods for formula (11) or (12). Therefore, we can acquire the final definition of a signal X (D-dimensional feature vector of each sample on the graph structure data) and filter g_θ in the Fourier domain. The first method is named TGCN-3 and is denoted as

$$Z = g_\theta(L) * \boldsymbol{X} = \boldsymbol{\theta}\,[(\boldsymbol{I}_N - L)]\boldsymbol{X} = [\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)}]\,\boldsymbol{X}\boldsymbol{\theta}. \quad (13)$$

The second method is named TGCN-4 and is denoted as (see (14)) . The third is a renormalisation method that is named TGCN and is denoted as $\boldsymbol{I}_N + 2\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)} \to \tilde{\boldsymbol{D}}^{-(1/2)}\tilde{\boldsymbol{B}}\tilde{\boldsymbol{D}}^{-(1/2)}$ with $\tilde{\boldsymbol{B}} = 2\boldsymbol{A} + \boldsymbol{I}_N$ and $\widetilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{B}}_{ij}$:

$$Z = g_\theta(L) * \boldsymbol{X} = \boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)}\tilde{\boldsymbol{D}}^{-(1/2)}\tilde{\boldsymbol{B}}\tilde{\boldsymbol{D}}^{-(1/2)}\boldsymbol{X}\boldsymbol{\theta} \quad (15)$$

where $\boldsymbol{\theta}$ is the weight matrix of one layer, $\boldsymbol{X}$ is the feature information matrix of the graph structure data and $\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)}$, $2\,(\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)})^2$ and $\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)}\tilde{\boldsymbol{D}}^{-(1/2)}\tilde{\boldsymbol{B}}\tilde{\boldsymbol{D}}^{-(1/2)}$ are the two-order localised structure information matrices that represent the manifold of the data. Z is the one convolution layer output of the model, which is the data information (structure information and feature information) that is assimilated from each layer of the neural network. In addition, we can use higher order polynomials of Laplacian. However, with the increase of the order number, the computation complexity of the model also increases. The optimisation of higher order polynomials is difficult.

First of all, as a pre-processing process calculation of $\widetilde{\boldsymbol{B}} = \boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)}$, $\tilde{\boldsymbol{B}} = 2(\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)})^2$, or $\widetilde{\boldsymbol{B}} = \boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)}\tilde{\boldsymbol{D}}^{-(1/2)}\tilde{\boldsymbol{B}}\tilde{\boldsymbol{D}}^{-(1/2)}$ is done. Thus, we can get the definition of the first convolution layer, i.e.

$$\boldsymbol{H}^{(1)} = \mathrm{Relu}\left(\tilde{\boldsymbol{B}}\boldsymbol{X}\boldsymbol{W}^{(0)}\right) \quad (16)$$

Here, $\tilde{\boldsymbol{B}}$ is an $N \times N$ symmetric matrix, which is represented as the localised structure information of each sample. In addition, we use the zero–one weighting method to construct adjacency matrix $\boldsymbol{A}$. $\boldsymbol{X}$ is an $N \times D$ matrix, which is used to represent the graph structure data feature information (N represents the number of samples and D is the dimension of each sample), i.e. the feature information of raw dataset. $\boldsymbol{W}^{(0)}$ is the first layer weight matrix. Relu is the nonlinear activation function of the first layer, which is no longer a linear combination of inputs and can approximate arbitrary functions. $\boldsymbol{H}^{(1)}$ represents the extracted sample feature matrix by the first layer of the TGCN, which is an $N \times M$ matrix (M is the number of the hidden neurons).

3.2 Second convolution layer of TGCN

The process of the second convolution layer is similar to the first layer. First, we do a pre-processing process with

$$\widetilde{\boldsymbol{B}} = \boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)}, \tilde{\boldsymbol{B}} = 2\,(\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)})^2 \quad (17)$$

or.

$$\tilde{\boldsymbol{B}} = \boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)}\tilde{\boldsymbol{D}}^{-(1/2)}\tilde{\boldsymbol{B}}\tilde{\boldsymbol{D}}^{-(1/2)}. \quad (18)$$

then, we take the output of the first layer as the input of the second layer. Finally, we acquire the formula of the second layer and is denoted as

$$Z = \boldsymbol{H}^{(2)} = \tilde{\boldsymbol{B}}\,\boldsymbol{H}^{(1)}\boldsymbol{W}^{(1)} \quad (19)$$

Here, $\boldsymbol{H}^{(1)}$ is the characteristic matrix of the first convolution layer. $\boldsymbol{W}^{(1)}$ is the weight matrix in the second layer. We can obtain the final features through the calculation of second layer, which is an $N \times E$ matrix (the number of E is the same as the class of the dataset). In this paper, we only use the two-layer TGCN and the TGCN also achieves good performance. A deeper TGCN model can be built through the repeated process of the above.

3.3 Softmax layer of TGCN

Through the two convolution layers, we can extract the effective identification feature of dataset. We put the extracted features into the classifier for classification. In addition, we use the Softmax classifier in the paper. The Softmax classifier is a model with a polynomial distribution, which can be divided into multiple mutually exclusive categories. It can be denoted as

$$f(Z_j) = \frac{\mathrm{e}^{Z_j}}{\sum_{i=1}^{n} \mathrm{e}^{Z_i}} \quad (20)$$

The output of the Softmax layer is also an $N \times E$ matrix and the value ranges from 0 to 1. Each value in the matrix belongs to the probability of each class. In the back propagation of neural networks model, a loss function is required. This loss function actually represents the error between the real value and the estimated value of the network model. In order to get the best classification results of the model, in this paper, we use the cross entropy loss function to the optimal hyper-parameter:

$$C = -\sum_k y_k \log Z_K \quad (21)$$

Here, Z_K indicates the output value of the kth neuron. y_K represents the true value corresponding to the kth neuron and takes a value of zero or one.

$$Z = g_\theta(L) * \boldsymbol{X} = \theta\,[2\,(L - \boldsymbol{I}_N)^2]\,\boldsymbol{X} = [2\,(\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)})^2]\,\boldsymbol{X}\boldsymbol{\theta} \quad (14)$$

Table 1 Experiment datasets

Dataset	Publications	Categories	Words
Citeseer	3327	6	3703
Cora	2708	7	1433
PubMed	19,717	3	500

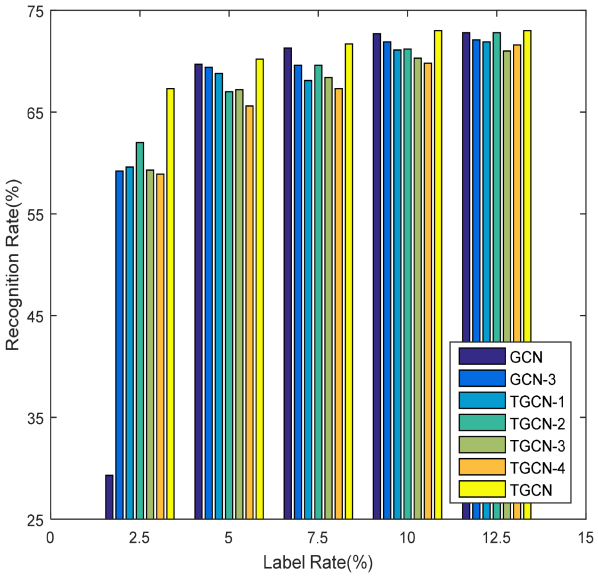

Fig. 3 *Recognition accuracy for all class of Citeseer database*

In the course of training, TGCN constantly updates the parameters through the number of the loss values to improve forecasting effect. There is no back propagation process in the verification and testing phase of the TGCN network.

4 Experiment

In this experiment part, we conduct substantial experiments on citation network dataset including Citeseer, Cora, and PubMed [26] to prove the effectiveness of the proposed algorithm for document classification. First of all, we introduce the experiment dataset. Then, we describe the experimental set-up that is used. Finally, we show the experimental results.

4.1 Citation network dataset

The Citeseer dataset [27] has a total of 3327 scientific publications and is divided into six categories, i.e. Agents, AI, DB, IR, ML, and HCI. The features of each publication represent different words and show the absence or presence of the corresponding word through zero and one. It consists of 4732 citation relationships for all publications and 3703 words for each publication.

The Cora dataset [28] is composed of 2708 scientific publications and contains seven classes, i.e. case-based, genetic algorithms, neural networks, probabilistic methods, reinforcement learning, rule learning, and theory. It has 5429 citation relationships between scientific publications and 1433 words for each publication.

The PubMed dataset [29] consists of totally 19,717 scientific publications collected from diabetes. It contains three categories, i.e. diabetes mellitus experimental, diabetes mellitus type 1 and diabetes mellitus type 2. Each publication is represented by the word vector. It is composed of 44,338 citation relationships and 500 feature vectors for each book. The experiment datasets are summarised in Table 1.

4.2 Experiment set-up

For the Citeseer, Cora, and PubMed dataset, we select 1000 scientific publications for the testing set, 500 publications as validation set and the remaining samples as training set. In our semi-supervised document classification experiments, the validation set and testing set are all labelled data. For the Citeseer and Cora datasets, we use a certain percentage labelled samples in the training data, which are 2.5, 5, 7.5, 10, 12.5, 15, 17.5, and 20%. For PubMed dataset, we take a certain percentage training data (i.e. 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9%) as labelled data. In addition, we conduct multiple experiments to ensure the correctness of the results (15, 17.5, 20, 0.7, 0.8, and 0.9%). The remaining samples of the training set are all unlabelled samples.

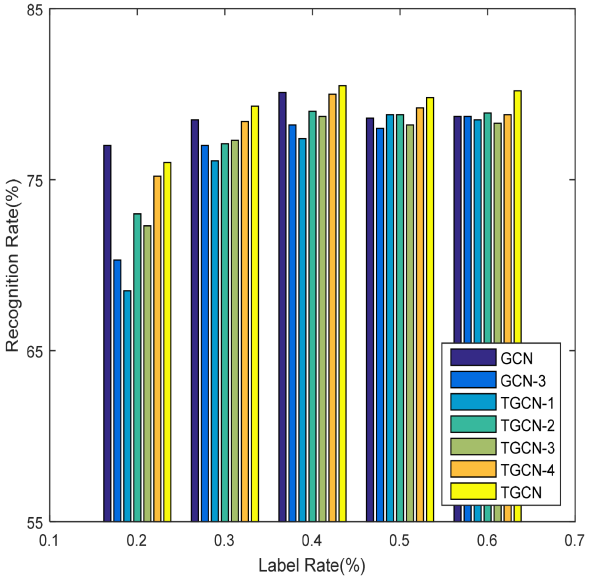

Fig. 4 *Recognition accuracy for all class of PubMed database*

We use the learning rate of 0.01 with Adam optimisation algorithm (a method for stochastic optimisation) [30] to update neural network weights iteratively. Adam calculates independent adaptive learning rates for different parameters by calculating the one-order and two-order moment estimates of the gradient. The maximum iteration times of the training model are 200. In addition, if the cross-entropy loss values of the model in the validation set are not increased for ten consecutive times, the two-layer TGCN will stop training automatically. We use the Xavier initialisation method to initialise weights, which has a detailed introduction in [31]. To prevent the overtraining of the model, we use the $L2$ regularisation method in the experiment. The $L2$ regularisation term coefficient is 5×10^{-4}. We also use other hyper-parameter including the number of hidden neurons and the dropout rate of the model. Other hyper-parameters are as follows: Citeseer: 0.5 (dropout rate) and 32 (hidden neurons); Cora: 0.5 (dropout rate) and 16 (hidden neurons); and PubMed: 0.7 (dropout rate) and 32 (hidden neurons).

4.3 Experiment results

In the experiment, the baseline methods are GCN and third-order GCNs (GCN-3), manifold regularisation (ManiReg) [32], semi-supervised embedding (SemiEmb) [33], label propagation (LP) [34], DeepWalk [11], iterative classification algorithm (ICA) [35], Planetoid [36]. In [14], GCN-3 is denoted as

$$g_\theta(L) * \boldsymbol{X} = \theta_0 \boldsymbol{X} + \theta_1 (L - \boldsymbol{I}_N)\boldsymbol{X} + \theta_2 [2(L - \boldsymbol{I}_N)^2 - \boldsymbol{I}_N] \times \boldsymbol{X} + \theta_3 [4(L - \boldsymbol{I}_N)^3 - 3(L - \boldsymbol{I}_N)]\boldsymbol{X}. \quad (22)$$

First, in this section, the comparison of the GCN and GCN-3 with the different variants of the TGCN on citation network dataset is shown, i.e. formulae (9) GCN, (10) TGCN-1, (11) or (12) TGCN-2, (13) TGCN-3, (14) TGCN-4, (15) TGCN, and (22) GCN-3.

The experimental results for all classes of citation network dataset are summarised in Figs. 3–5 and Tables 2–4. For each figure, the x-axis expresses the different label rate and the y-axis expresses the recognition rate of all classes. Figs. 6–11 show the recognition rate of each class of the different model. For each class,

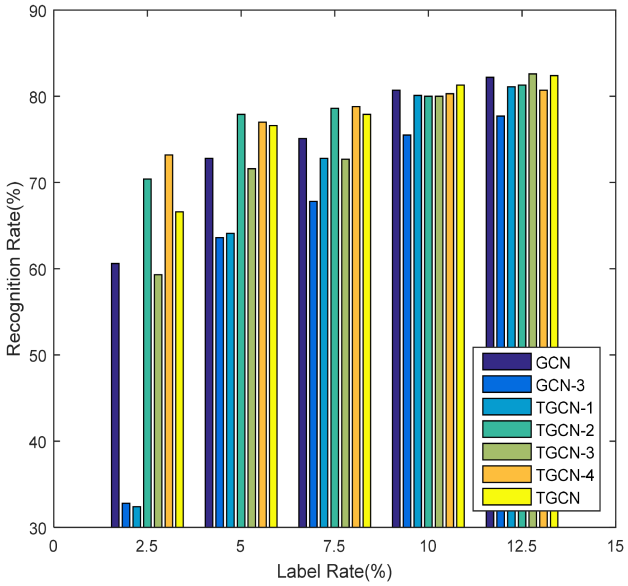

Fig. 5 *Recognition accuracy for all class of Cora database*

Table 2 Recognition accuracy for all classes of Citeseer database

Dataset label rate	15%	17.5%	20%
GCN	72.28	72.58	73.44
GCN-3	71.68	71.84	72.46
TGCN-1	71.52	73.52	73.8
TGCN-2	70.66	71.72	72.24
TGCN-3	71.22	71.94	72.32
TGCN-4	69.92	70.72	70.82
TGCN	73.12	73.62	74.02

Table 3 Recognition accuracy for all classes of PubMed database

Dataset label rate	0.7%	0.8%	0.9%
GCN	80.86	80.94	81.12
GCN-3	79.32	79.6	79.8
TGCN-1	78.08	79.44	79.68
TGCN-2	80.28	80.7	81.3
TGCN-3	80.26	80.56	80.94
TGCN-4	79.12	80.14	80.98
TGCN	81.34	81.54	81.98

the y-axis is the experimental result of each class. The experiment aims to prove the effectiveness for the derivation process of the novel layerwise linear model.

As shown in Fig. 3 and Table 2, as the label rate increases, the GCN, GCN-3, and the different variants of the TGCN all get a higher recognition rate on Citeseer dataset. In addition, TGCN acquires the highest recognition rate, no matter how the label rate changes. Compared with GCN and GCN-3 on Citeseer dataset, Figs. 6 and 7 demonstrate that TGCN also achieves the best recognition rate in most classes.

Fig. 4 and Table 3 show the recognition rate of GCN, GCN-3, and the different variants of the TGCN for all classes of the PubMed database. From the experimental results, we can know that the TGCN method is the best in most cases. Moreover, as shown in Fig. 8, GCN, GCN-3, and the different variants of the TGCN implement the highest recognition rate of each class.

Fig. 5 and Table 4 prove that the TGCN method outperforms GCN and GCN-3 on the Cora dataset in the different label rate cases. As the Cora dataset is quite small, it is impossible to achieve the best performance of TGCN method in contrast to certain variants of TGCN. However, the TGCN still has a high recognition rate. With an increase in the label rate, the recognition results of all classes also increase. In contrast with GCN and GCN-3, Figs. 9–11 reveal that TGCN also gains the highest performance under most classes.

Table 4 Recognition accuracy for all classes of Cora database

Dataset label rate	15%	17.5%	20%
GCN	80.26	80.44	80.68
GCN-3	76.76	80	80.38
TGCN-1	79.84	81.4	82.6
TGCN-2	81.2	81.72	82.42
TGCN-3	78.82	80.64	82.3
TGCN-4	80.76	81.86	82.84
TGCN	81.62	82.94	83.64

In addition, the comparison of different semi-supervised methods is summarised in Table 5. Report numbers represent the recognition rate in percentage. From Table 5, we can find out that, compared with other methods, TGCN gets the best performance in recognition rate and speed.

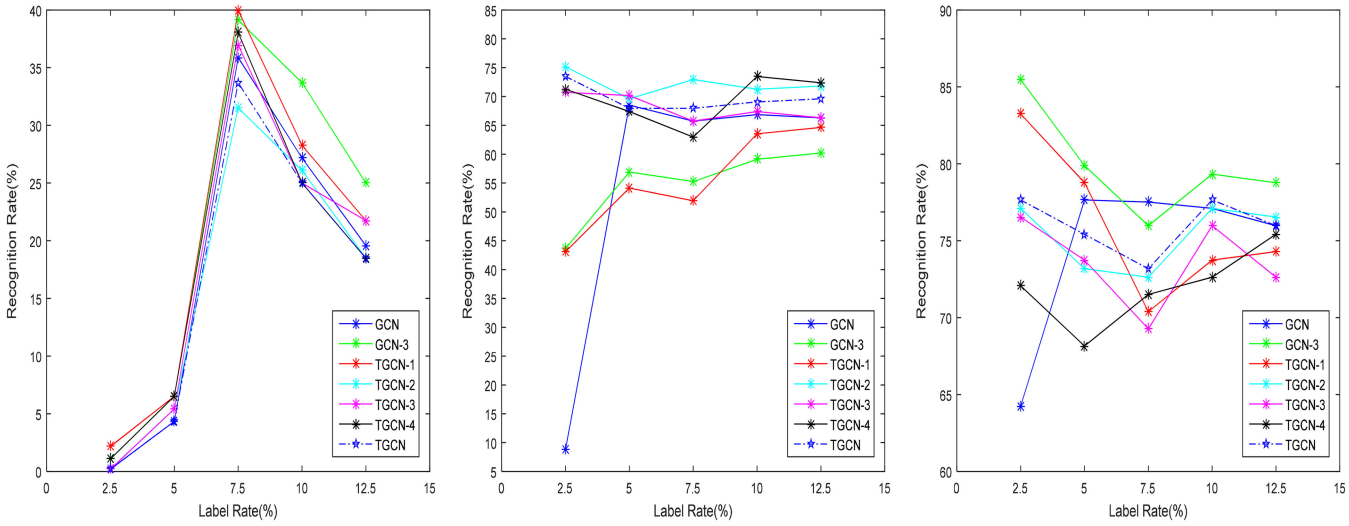

Fig. 6 *Recognition accuracy for each class of Citeseer database including Agents, AI, DB. Each subfigure corresponds to a single class*

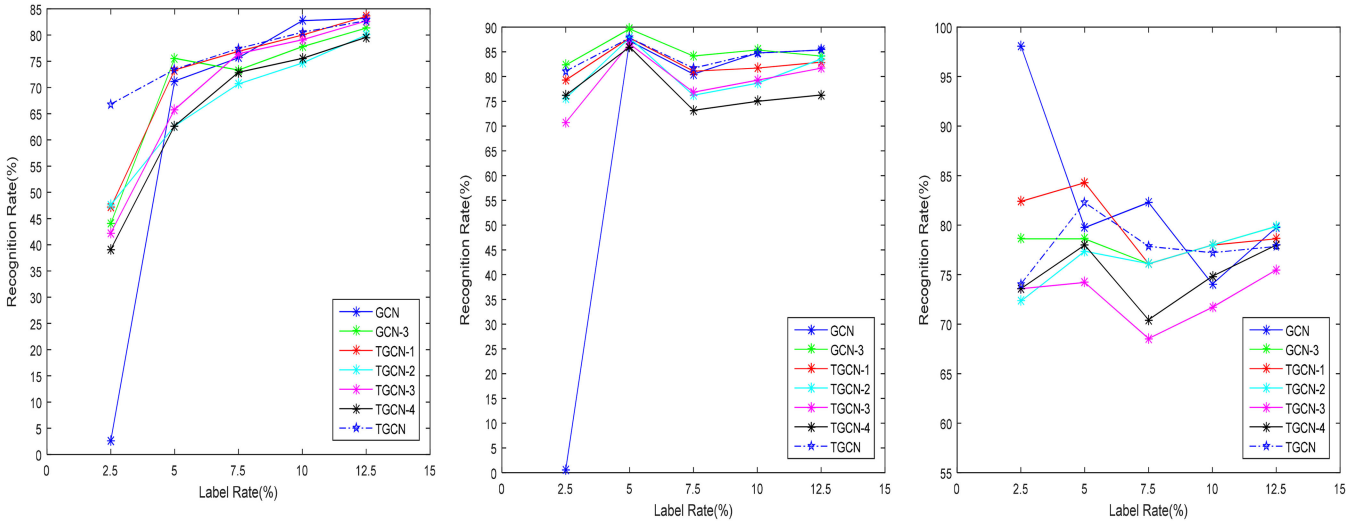

Fig. 7 *Recognition accuracy for each class of Citeseer database including IR, ML, HCI. Each subfigure corresponds to a single class*

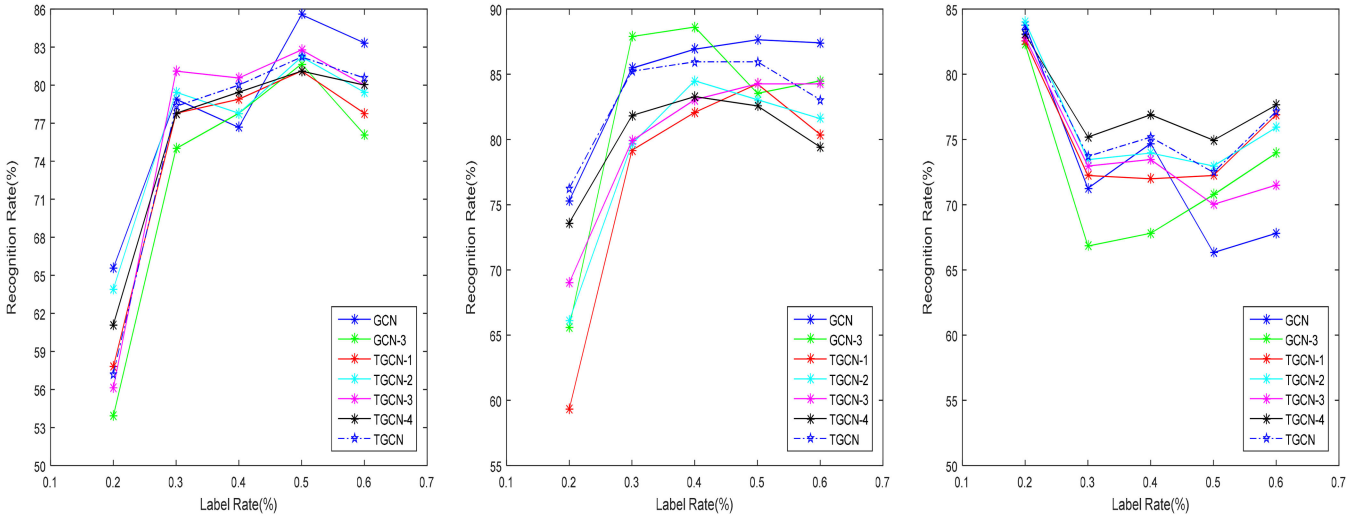

Fig. 8 *Recognition accuracy for each class of PubMed database, including diabetes mellitus experimental, diabetes mellitus type 1, diabetes mellitus type 2. Each subfigure corresponds to a single class*

5 Conclusion

As a result of the existence of numerous data in our real life that does not have a regular spatial structure, at the present time, the graph structure data has become the object of the extensive research. How to extract valid sample information is central for deep learning. The sample information is composed of the feature information and the structure information. There have existed many effective methods to extract the feature information of the graph structure data until now. However, how to express a rich and localised structure information is still a challenging question. To address the problem, in this paper, we put forward a novel

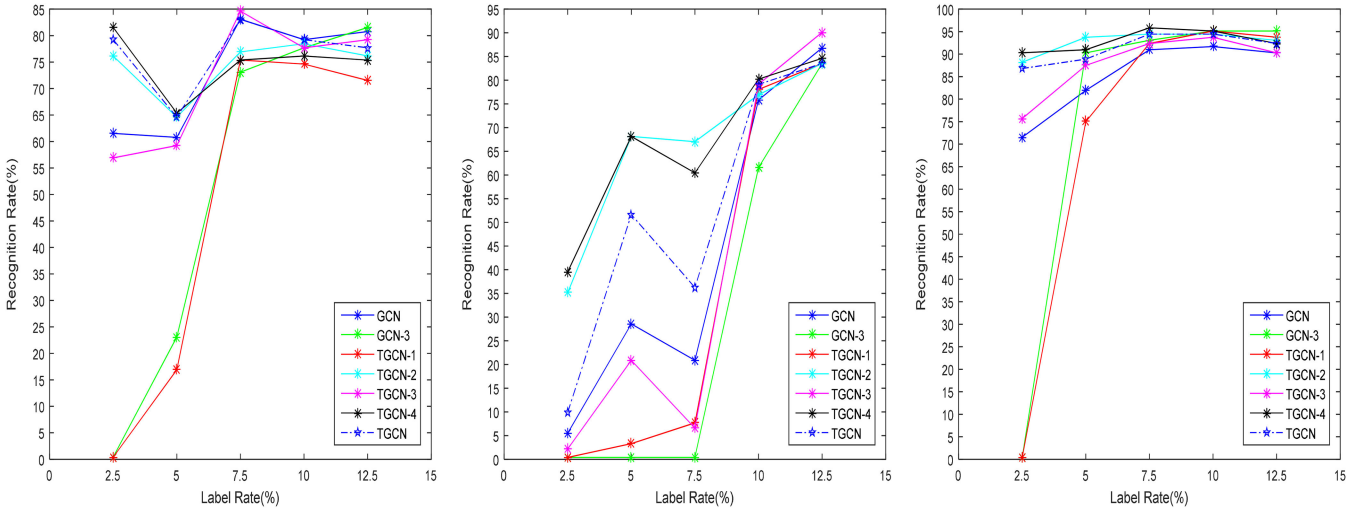

Fig. 9 *Recognition accuracy for each class of Cora database, including case based, genetic algorithms, neural networks. Each subfigure corresponds to a single class*

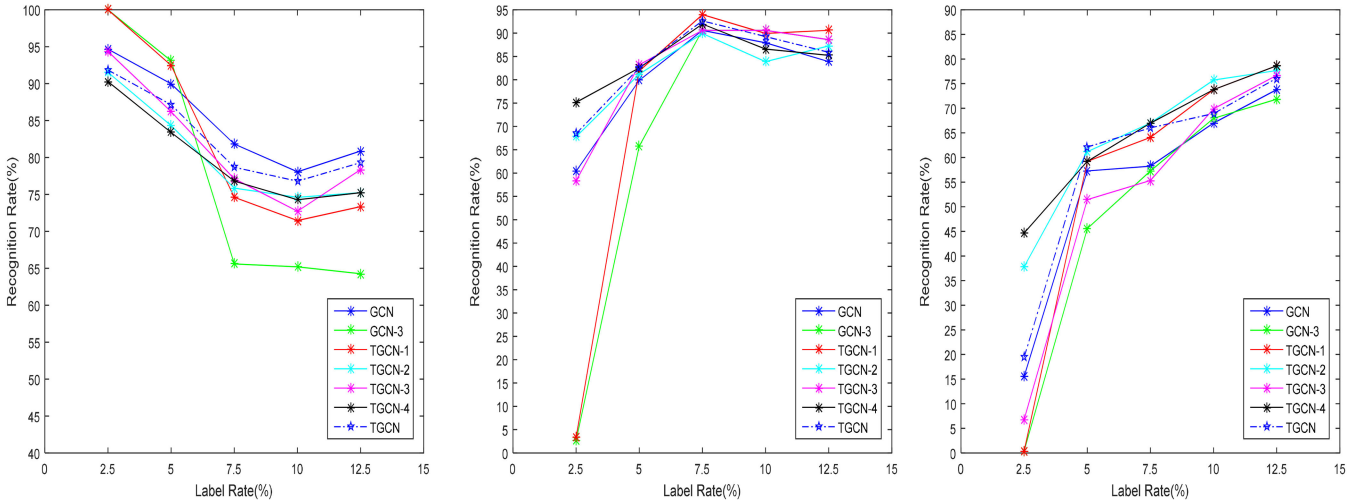

Fig. 10 *Recognition accuracy for each class of Cora database including probabilistic methods, reinforcement learning, rule learning. Each subfigure corresponds to a single class*

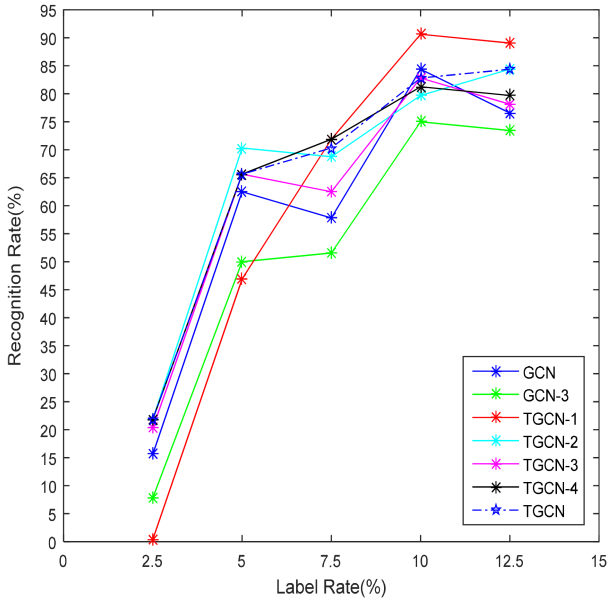

Fig. 11 *Recognition accuracy for theory class of Cora database*

networks model, two-order GCNs. Specifically, TGCN utilises a two-order polynomial of the spectral graph convolution to express the localised manifold structure information of the graph structure data. In addition, we continuously optimise the definition of two-order approximation for spectral convolution on graph data. Substantial experimental results demonstrate that superior

Table 5 Comparison of different methods

Method label rate	Citeseer 6.5%	Cora 11.5%	PubMed 0.3%
ManiReg	60.1	59.5	70.7
SemiEmb	59.6	59.0	71.1
LP	45.3	68.0	63
DeepWalk	43.2	67.2	65.3
ICA	69.1	75.1	73.9
Planetoid	64.7(26 s)	75.7(13 s)	77.2(25 s)
GCN	70.3(7 s)	81.5(4 s)	79.0(38 s)
TGCN	71.4(3 s)	82(2 s)	79.5(14 s)

recognition accuracy can be achieved with TGCN model in the citation network datasets, compared with the existing methods including GCN and Planetoid.

6 Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61671480, in part by the Fundamental Research Funds for the Central Universities, China University of Petroleum (East China) under Grant 18CX07011A and YCX2019080.

7 References

[1] Wang, Q., Chen, M., Nie, F., *et al.*: 'Detecting coherent groups in crowd scenes by multiview clustering', *IEEE Trans. Pattern Anal.*, 2018, doi: 10.1109/TPAMI.2018.2875002, to be published

[2] Wang, Q., Wan, J., Li, X.: 'Robust hierarchical deep learning for vehicular management', *IEEE Trans. Veh. Technol.*, 2019, **68**, (5), pp. 4148–4156

[3] Meng, S., Dou, W., Zhang, X., *et al.*: 'KASR: a keyword-aware service recommendation method on mapreduce for big data applications', *IEEE Trans. Parall. Distr.*, 2014, **25**, (12), pp. 3221–3231

[4] Dou, W., Zhang, X., Liu, J., *et al.*: 'HireSome-II: towards privacy-aware cross-cloud service composition for big data applications', *IEEE Trans. Parall. Distr.*, 2015, **26**, (2), pp. 455–466

[5] Rabiner, L.R.: 'A tutorial on hidden Markov models and selected applications in speech recognition', *Proc. IEEE*, 1989, **77**, (2), pp. 257–286

[6] Graves, A., Mohamed, A., Hinton, G.: 'Speech recognition with deep recurrent neural networks'. IEEE Conf. Acoustics, Speech and Signal Processing, Vancouver, Canada, May, 2013, pp. 6645–6649

[7] Keys, R.: 'Cubic convolution interpolation for digital image processing', *IEEE Trans. Acoust. Speech Signal Process.*, 1981, **29**, (6), pp. 1153–1160

[8] Sun, T., Neuvo, Y.: 'Detail-preserving median based filters in image processing', *Pattern Recogn. Lett.*, 1994, **15**, (4), pp. 341–347

[9] Collobert, R., Weston, J.: 'A unified architecture for natural language processing: deep neural networks with multitask learning'. Proc. 25th Int. Conf. Machine Learning, Helsinki, Finland, July 2008, pp. 160–167

[10] Nasukawa, T., Yi, J.: 'Sentiment analysis: capturing favorability using natural language processing'. Proc. 2nd Int. Conf. on Knowledge Capture, Sanibel Island, FL, USA, October 2003, pp. 70–77

[11] Perozzi, B., Al-Rfou, R., Skiena, S.: 'Deepwalk: online learning of social representations'. Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, San Francisco, USA, August 2014, pp. 701–710

[12] Tang, J., Qu, M., Wang, M., *et al.*: 'Line: large-scale information network embedding'. Proc. 24th Int. Conf. World Wide Web Int. World Wide Web Conf. Steering Committee, May 2015, pp. 1067–1077

[13] Atwood, J., Towsley, D.: 'Diffusion-convolutional neural networks'. Advances in Neural Information Processing Systems, Barcelona, Spain, December 2016, pp. 1993–2001

[14] Kipf, T.N., Welling, M.: 'Semi-supervised classification with graph convolutional networks', Int. Conf. Learning Representations, Toulon, France, April 2017

[15] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., *et al.*: 'Convolutional networks on graphs for learning molecular fingerprints'. Advances in Neural Information Processing Systems, December 2015, pp. 2224–2232

[16] Turaga, S.C., Murray, J.F., Jain, V., *et al.*: 'Convolutional networks can learn to generate affinity graphs for image segmentation', *Neural Comput.*, 2010, **22**, (2), pp. 511–538

[17] Niepert, M., Ahmed, M., Kutzkov, K.: 'Learning convolutional neural networks for graphs'. Int. Conf. Machine Learning, Long Beach, California, June 2016, pp. 2014–2023

[18] Ktena, S. I., Parisot, S., Ferrante, E., *et al.*: 'Distance metric learning using graph convolutional networks: application to functional brain networks'. Int. Conf. Medical Image Computing and Computer-Assisted Intervention, Quebec City, Canada, September 2017, pp. 469–477

[19] Belkin, M., Niyogi, P.: 'Towards a theoretical foundation for Laplacian-based manifold methods'. Int. Conf. Computational Learning Theory, June 2005, pp. 486–500

[20] Von Luxburg, U.: 'A tutorial on spectral clustering', *Stat. Comput.*, 2007, **17**, (4), pp. 395–416

[21] Balaban, A.T.: 'Applications of graph theory in chemistry', *J. Chem. Inf. Model.*, 1985, **25**, (3), pp. 334–343

[22] Hammond, D.K., Vandergheynst, P., Gribonval, R.: 'Wavelets on graphs via spectral graph theory', *Appl. Comput. Harmon. A*, 2011, **30**, (2), pp. 129–150

[23] Defferrard, M., Bresson, X., Vandergheynst, P.: 'Convolutional neural networks on graphs with fast localized spectral filtering'. Advances in Neural Information Processing Systems, Barcelona, Spain, December 2016, pp. 3844–3852

[24] Wang, Q., Gao, J., Yuan, Y.: 'A joint convolutional neural networks and context transfer for street scenes labeling', *IEEE Trans. Intell. Transp.*, 2018, **19**, (5), pp. 1457–1470

[25] Wang, Q., Gao, J., Yuan, Y.: 'Embedding structured contour and location prior in siamesed fully convolutional networks for road detection', *IEEE Trans. Intell. Transp.*, 2018, **19**, (1), pp. 230–241

[26] Sen, P., Namata, G., Bilgic, M., *et al.*: 'Collective classification in network data', *AI Mag.*., 2008, **29**, (3), p. 93

[27] Poon, H., Domingos, P.: 'Joint inference in information extraction'. AAAI Conf. Artificial Intelligence, Vancouver, British, July 2007, pp. 913–918

[28] Cabanes, C., Grouazel, A., Schuckmann, K.V., *et al.*: 'The CORA dataset: validation and diagnostics of in-situ ocean temperature and salinity measurements', *Ocean Sci.*, 2013, **9**, (1), pp. 1–18

[29] Bilgic, M., Licamele, L., Getoor, L., *et al.*: 'D-dupe: An interactive tool for entity resolution in social networks'. Visual Analytics Science and Technology, October 2006, pp. 43–50

[30] Kinga, D., Adam, J.B.: 'A method for stochastic optimization'. Int. Conf. Learning Representations, San Diego, CA, USA, 2015

[31] Glorot, X., Bengio, Y.: 'Understanding the difficulty of training deep feedforward neural networks'. Proc. 13th Int. Conf. Artificial Intelligence and Statistics, March 2010, pp. 249–256

[32] Belkin, M., Niyogi, P., Sindhwani, V.: 'Manifold regularization: A geometric framework for learning from labeled and unlabeled examples', *J Mach. Learn. Res.*, 2006, **7**, pp. 2399–2434

[33] Weston, J., Ratle, F., Mobahi, H., *et al.*: 'Deep learning via semi-supervised embedding'. Neural Networks: Tricks of the Trade, Edinburgh, Scotland, 2012, pp. 639–655

[34] Zhu, X., Ghahramani, Z., Lafferty, J. D.: 'Semi-supervised learning using Gaussian fields and harmonic functions'. Proc. 20th Int. Conf. Machine Learning, Edinburgh, Scotland, June 2003, pp. 912–919

[35] Lu, Q., Getoor, L.: 'Link-based classification'. Proc. 20th Int. Conf. on Machine Learning, Edinburgh, Scotland, June 2003, pp. 496–503

[36] Yang, Z., Cohen, W. W., Salakhutdinov, R.: 'Revisiting semi-supervised learning with graph embeddings', Int. Conf. Machine Learning, New York City, NY, USA, June 2016