International Journal of Machine Learning and Cybernetics (2019) 10:3115–3127
https://doi.org/10.1007/s13042-019-01003-7

ORIGINAL ARTICLE

PKGCN: prior knowledge enhanced graph convolutional network for graph-based semi-supervised learning

Shaowei Yu[1,2] · **Xuebing Yang**[1] · **Wensheng Zhang**[1]

Received: 4 March 2019 / Accepted: 19 August 2019 / Published online: 27 August 2019

Abstract
Graph is a widely existed data structure in many real world scenarios, such as social networks, citation networks and knowledge graphs. Recently, Graph Convolutional Network (GCN) has been proposed as a powerful method for graph-based semi-supervised learning, which has the similar operation and structure as Convolutional Neural Networks (CNNs). However, like many CNNs, it is often necessary to go through a lot of laborious experiments to determine the appropriate network structure and parameter settings. Fully exploiting and utilizing the prior knowledge that nearby nodes have the same labels in graph-based neural network is still a challenge. In this paper, we propose a model which utilizes the prior knowledge on graph to enhance GCN. To be specific, we decompose the objective function of semi-supervised learning on graphs into a supervised term and an unsupervised term. For the unsupervised term, we present the concept of local inconsistency and devise a loss term to describe the property in graphs. The supervised term captures the information from the labeled data while the proposed unsupervised term captures the relationships among both labeled data and unlabeled data. Combining supervised term and unsupervised term, our proposed model includes more intrinsic properties of graph-structured data and improves the GCN model with no increase in time complexity. Experiments on three node classification benchmarks show that our proposed model is superior to GCN and seven existing graph-based semi-supervised learning methods.

Keywords Graph convolutional network · Semi-supervised learning · Prior knowledge · Node classification

1 Introduction

As a universal language for describing complex data and systems, graphs exist widely in the real world, such as social network [15], protein-protein interaction network [10] and traffic network [24, 40]. Analysis on graphs can bring us more insights and implicit information of data by utilizing the relations and interactions among the components [13]. One of the most important problems in graph analysis is classifying nodes of a graph with only a small portion of labeled nodes and the graph structure. In the context of machine learning, this problem is framed as graph-based semi-supervised classification [19]. The problem is worth studying since many machine learning models require a large amount of labeled data which is hard and expensive to obtain.

Conventionally, a large number of graph-based semi-supervised learning algorithms define the objective function as the weighted sum of supervised and unsupervised loss [2, 38, 42, 44]. For example, graph Laplacian regularization is typically used as the unsupervised loss, which is based on the assumption that connected nodes in graph are likely to have the same label [39]. These algorithms are efficient since the distribution of nodes label is constrained to be in accord with the graph structure. However, edges in graph do not only indicate nodes similarity, but also contain additional intrinsic information about the graph structure, which does not be fully exploited in graph Laplacian-based algorithms.

Since convolutional neural networks (CNNs) [22] have achieved state-of-the-art performance in a broad range of tasks from regular Euclidean domains like image [20],

✉ Shaowei Yu
yushaowei2017@ia.ac.cn

Xuebing Yang
yangxuebing2013@ia.ac.cn

Wensheng Zhang
zhangwenshengia@hotmail.com

[1] Institute of Automation, Chinese Academy of Sciences, 95 Zhongguancun East Road, 100190 Beijing, China

[2] University of Chinese Academy of Sciences, Beijing, China

acoustics [34] and natural language [17], generalizing CNNs to non-Euclidean domains is proven to be a promising direction for graph analysis. Recently developed Graph Convolutional Neural Networks (GCNNs) [5, 8, 19] have achieved remarkable success and been widely applied in graph analysis. As for graph-based semi-supervised learning, Graph Convolutional Network (GCN) [19] utilizes convolution operation defined on graphs to extract features. The GCN model overcomes the shortcomings of the conventional methods by propagating features based on the graph structure through multiple layers. On a number of benchmarks of graph-based semi-supervised classification, GCN outperforms state-of-the-art methods both in accuracy and efficiency. MoNet [30] and Graph Attention Network (GAT) [37] further improve GCN through introducing more complex propagation mechanisms. The propagation and updating rule for GNNs are straightforward and GNNs have achieved the state-of-the-art performance in many tasks. However, these models have similar limits with many CNN-based models that the working mechanism such as the representation properties and capacities have not been made clear. The design of model structure is mostly based on intuition, heuristic method and experiment results. Besides, a lot of effort is required for parameter tuning due to lack of prior knowledge in the optimization objective.

Our basic motivation is based on homophily theory [28] in sociology that two connected nodes in graph are more likely to share similar interests. The prior knowledge is adopted as the basic assumption in graph Laplacian-based methods [42, 44]. Nevertheless, no GCNNs have encoded the prior knowledge in the model to directly optimize. In [19], GCN is used as a powerful feature extractor and the loss function is totally supervised which only defined on labeled nodes. However, the lack of unsupervised information in the optimization objective leads to over-fit on the labeled data. In graph Laplacian based methods, the prior knowledge that similar nodes have the same labels is used as unsupervised information. Therefore, our idea is to use GCN as the feature extractor and include the unsupervised information in the optimization objective. To be specific, we define local inconsistency as the distance between the probability mass function (pmf) of two connected nodes. Then we introduce the inconsistency loss which is an unsupervised loss imposing a penalty on local inconsistency. Finally, we propose the new loss function for GCN as the ensemble of supervised loss and unsupervised loss. Compared with the conventional Laplacian-based methods, our proposal contains more structural information by using GCN-based structure. Besides, we further improved the original GCN model by encoding prior knowledge in the loss function.

In this paper, we propose Prior Knowledge enhanced Graph Convolutional Network (PKGCN) to overcome the shortcomings of both graph Laplacian regularization based methods and the GCN model. The main contributions of our work can be summarized as follows:

- We propose a new paradigm which combines the graph Laplacian-based methods and GCN to overcome the shortcomings from two kinds of methods.
- We introduce a new loss function of GCN for graph-based semi-supervised classification. In addition to labeled information, the loss function further incorporates prior knowledge on graphs and imposes a penalty on local inconsistency.
- Extensive experiments show that our method yields better results than the GCN model with no increase in time complexity and no particular modification of the GCN model structure.

The rest of this paper is organized as follows. In Sect. 2, the related works are briefly reviewed, including semi-supervised learning and graph-based neural network. In Sect. 3, we introduce notations and preliminaries. The proposed PKGCN is presented in details in sect. 4. The experiments and analysis are shown in Sect. 5. Finally, we give our conclusions in Sect. 6.

2 Related work

2.1 Semi-supervised learning

Semi-supervised learning [6, 43] aims at making use of a small amount of labeled data and a large amount of unlabeled data to learn a classifier. A common assumption in semi-supervised learning is the manifold assumption that data distributes in a manifold structure and nearby samples have similar output values. Currently, there are four paradigms in semi-supervised learning: generative methods, semi-supervised support vector machine (S3VM), graph-based methods and disagreement-based methods. Generative methods [11, 32] assume that all samples are generated from the same latent model. The missing labels are regarded as the missing parameters and can be estimated by EM algorithm. S3VM is a generalization of SVM and based on the assumption of low-density separation. The most famous method in this paradigm is the transductive vector machine (TSVM) [16]. Graph-based methods [2, 26, 38, 44] consider the relationships among data and map the data to a graph. Disagreement-based methods assume that different views of data are compatible and complementary. One of the common algorithms is co-training [3], which trains multiple classifiers and each classifier passes its most-confident samples to other classifiers.

The problem we consider in this paper is the graph-based semi-supervised classification. Methods can be roughly

divided into two categories. The first category uses a graph Laplacian regularization term in the loss function, which is based on the assumption that adjacent nodes in the graph tend to share the same labels. Therefore a large penalty will incur when data points with high proximity are predicted to have different labels. Various methods based on this assumption adopt variants of graph Laplacian in the loss function, including LP [44], ManiReg [2], SemiEmb [38] and ICA [26]. The graph Laplacian regularizer is efficient when the graph fits the task. However, edges could contain additional information except for proximity, which is not leveraged in graph Laplacian regularization based methods. The second category is based on graph embedding, which aims to obtain a low-dimensional vector for each node while preserving the graph structure [13]. The embeddings can be used as input features of downstream machine learning models. Inspired by skip-gram model [29], DeepWalk [33] learns node embeddings through predicting its context generated by random walk. Node2vec [14] improves DeepWalk by employing a biased random walk scheme. Planetoid [39] learns node embedding through jointly predicting the class label and the context of the node in the graph. However, these methods are not end-to-end as random walk sequences generation and parameters learning are optimized separately.

2.2 Graph-based neural networks

The most primitive graph-based neural network is Graph Neural Network (GNN) [35] which extends recurrent neural network for graph-structured data. To learn parameters, GNN needs to iteratively apply contraction maps until convergence to obtain node embedding, which restricts the model efficiency and ability. Gated Graph Neural Network (GGNN) [23] further improves GNN through introducing gate mechanism and thus removes the need for contraction maps. Besides GNN, Graph Convolutional Neural Networks (GCNNs) are another kinds of graph-based neural networks which generalizes CNNs to graph domain. According to the difference of convolutional operations, GCNNs can be classified into two types [4]: spatial methods and spectral methods. Spatial methods consider convolution as a patch operator and operate on spatially neighbors to propagate node information [1, 9, 30, 31]. Spectral methods generalize Convolutional Theorem [27] on graphs as the multiplication of a graph signal and a spectral filter [5, 36]. However, these models require computing the eigenvectors of graph Laplacian matrix, which is computationally expensive for large graphs. To improve efficiency, ChebyNet [8] simplifies the spectral filter as a K^{th}-order truncated Chebyshev polynomial expansion of the diagonal matrix of Laplacian eigenvalue. GCN [19] further reduces the computational cost of ChebyNet by limiting $K = 1$ and using renormalization trick. Through simplification and approximation, the spectral filter of GCN is spatially localized and only uses the first-order neighbors. However, our proposed model is different from previous work as we introduce prior knowledge on graphs in the optimization objective of GCN through combining supervised term and unsupervised term.

3 Notations and preliminary

Given the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ is a set of $N = |\mathcal{V}|$ nodes and $\mathcal{E} = \{(v_i, v_j) \mid v_i, v_j \in \mathcal{V}\}$ is a set of $M = |\mathcal{E}|$ edges. For a given node v_i, we use $N(v_i) = \{v_j \mid (v_i, v_j) \in \mathcal{E}\}$ to denote the set of nodes directly connected to node v_i. We use $\mathbf{A} \in \mathbb{R}^{N\times N}$ to denote the adjacency matrix. $A_{ij} = 1$ if $(i,j) \in \mathcal{E}$, otherwise $A_{ij} = 0$. The degree matrix $\mathbf{D} \in \mathbb{R}^{N\times N}$ is a diagonal matrix $D_{ii} = \sum_j A_{ij}$. We denote $\mathbf{X}_i \in \mathbb{R}^D$ as the feature vector of node v_i. $\mathbf{X} \in \mathbb{R}^{N\times D}$ as the feature matrix of all nodes in the graph. Let $\mathcal{V}_L$ be the set of labeled nodes and $\mathcal{V}_U$ be the set of unlabeled nodes. $\mathcal{V}_L \cup \mathcal{V}_U = \mathcal{V}$. For each node in $\mathcal{V}_L$, the label is denoted as a one-hot vector $\mathbf{Y}_i \in \mathbb{R}^C$, where C is the number of classes. We use $\mathcal{Y}_L$ to denote the set of labels for nodes in $\mathcal{V}_L$. The problem of graph-based semi-supervised learning is to predict the label of nodes in $\mathcal{V}_U$ with $\mathbf{X}$, $\mathbf{A}$ and $\mathcal{Y}_L$.

GCN [19] generalizes and simplifies convolutional theorem [27] on graphs as follows:

$$\mathbf{g}_\theta \star \mathbf{x} \approx \theta(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{x}. \tag{1}$$

Here, $\mathbf{x} \in \mathbb{R}^N$ is signal defined on the nodes of the graph and $\mathbf{g}_\theta \in \mathbb{R}^N$ is spectral representation of the filter parameterized by $\theta \in \mathbb{R}$. Using renormalization trick $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and generalizing to signals with multiple input channels and filters, the propagation rule becomes:

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

Here, $\mathbf{H}^{(l)} \in \mathbb{R}^{N\times D^{(l)}}$ is the input and $\mathbf{H}^{(l+1)} \in \mathbb{R}^{N\times \mathbf{D}^{(l+1)}}$ is the output of l^{th} layer of GCN. $\mathbf{H}^{(0)} = \mathbf{X}$. $\sigma(\cdot)$ denotes the activation function. $\mathbf{W}^{(l)}$ is the trainable parameter of *lth* layer.

For semi-supervised node classification, [19] used a two-layer GCN. Given the nodes features matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$ as input, the GCN model applies a *softmax* classifier on the output:

$$\mathbf{Z} = f(\mathbf{X}, \mathbf{A}) = \text{softmax}\Big(\hat{\mathbf{A}}\ \text{ReLU}\Big(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}\Big)\boldsymbol{W}^{(1)}\Big). \tag{3}$$

Here, $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is calculated in the pre-processing step. $\mathbf{W}^{(0)} \in \mathbb{R}^{D\times H}$ is the parameter of the input-to-hidden layer with H feature maps. $\mathbf{W}^{(1)} \in \mathbb{R}^{H\times C}$ is the parameter of the hidden-to-output layer. ReLU is the activation function defined as $\text{ReLU}(x) = \max(x, 0)$. The *softmax* activation function is defined as $\text{softmax}(x_i) = \exp(x_i)/\sum_i \exp(x_i)$. $\mathbf{Z} \in \mathbb{R}^{N\times C}$ is the output the GCN model and C is the

dimension of the output features, which is equal to the number of classes. The *ith* row vector $\mathbf{Z}_i \in \mathbb{R}^C$ is the discrete predicted class pmf of node v_i over C classes. The loss function is the negative logarithmic likelihood defining over all labeled examples:

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{c=1}^{C} Y_{lc} \log Z_{lc}. \tag{4}$$

GCN works well since the graph convolution operation aggregates feature information from neighboring nodes and propagates it layer by layer. However, since the loss function is the negative logarithmic likelihood only defined on the labeled nodes, the GCN model lacks instructions for unlabeled nodes in the optimization objective, resulting in over-fitting on the labeled nodes.

4 The proposed method

In this Section, we first introduce the concept and the definition of local inconsistency. Next, We introduce how to calculate inconsistency loss with local inconsistency. Finally, we introduce the final model combining supervised loss and inconsistency loss. The pseudocode of the proposed PKGCN is presented in Algorithm 1.

4.1 Local inconsistency

The social correlation theory homophily [28] states that two connected users are likely to share similar interests. For example, in a citation network, nodes represent papers and edges represent citation relationships. Links usually exist between two papers in similar field. In our proposed model, we adopt the assumption that linked nodes are more likely to belong to the same class (as shown in Fig. 1, linked nodes form a densely connected cluster).

Under the assumption, two linked nodes should have high similarity in terms of pmf predicted by the model. Note that the pmf is defined as the probability that a discrete random variable X takes on a particular value x, that is, $P(X = x)$. We define local inconsistency on graphs as the distance of pmfs between two linked nodes. We give the formal definition of local inconsistency as follow:

Definition 1 *(local inconsistency)* Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph. $v_i \in \mathcal{V}$, $v_j \in \mathcal{V}$ are two nodes linked by edge $(v_i, v_j) \in \mathcal{E}$. $\mathbf{Z}_i \in \mathbb{R}^C$ and $\mathbf{Z}_j \in \mathbb{R}^C$ are the pmfs over C classes of v_i and v_j respectively. The local inconsistency of v_i and v_j is

$$I_{local}(v_i, v_j) = d(\mathbf{Z}_i, \mathbf{Z}_j), \tag{5}$$

where $d(\cdot, \cdot)$ is a distance measure of pmfs.

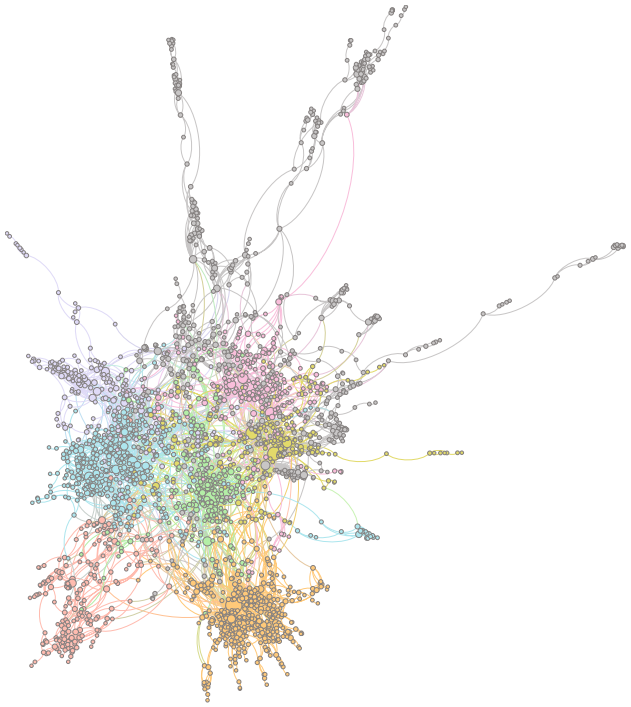


Fig. 1 Visualization of the Cora dataset. Nodes correspond to scientific publications and edges to citations. Marker color represents the groundtruth class of the node. Nodes belonging to the same class are more likely to connect with each other, forming a densely connected cluster. More details of the dataset are given in Sect. 5.1

Here, we provide two forms of distance measure function $d(\cdot, \cdot)$. The first one is the Euclidean distance, which is a normal distance measure in Euclidean vector space. Euclidean distance is defined as

$$d_{EUC}(P, Q) = \sqrt{\sum_{x \in \mathcal{X}} [P(x) - Q(x)]^2}. \tag{6}$$

Since the variables of the distance function are pmfs, we also use Jensen–Shannon divergence (JS divergence) [25] to measure the distance. JS divergence is a method of measuring the similarity between two pmfs. It is based on the Kullback–Leibler divergence [21], but it is symmetric and always has a finite value. Given two discrete pmfs P and Q defined on the same probability space, the JS divergence between P and Q is defined to be

$$d_{JS}(P, Q) = \sum_{x \in \mathcal{X}} \frac{1}{2} \left[P(x) \log \left(\frac{P(x)}{\frac{P(x)+Q(x)}{2}} \right) + Q(x) \log \left(\frac{Q(x)}{\frac{P(x)+Q(x)}{2}} \right) \right]. \tag{7}$$

The local inconsistency measures the similarity between two nodes' pmfs predicted by GCN [19]. High local inconsistency indicates low similarity, vice versa. Under the previous assumption, two nodes linked by an edge should have a high similarity between their predicted class pmfs, resulting in low local inconsistency.

4.2 Inconsistency loss

For a given node v_i, $N(v_i)$ is the set of nodes directly connected with v_i. $\mathbf{Z}_i \in \mathbb{R}^C$ is the class pmf of node v_i predicted by GCN model. The local inconsistency of node v_i and its neighbor $v_j \in N(v_i)$ can be computed using Eq. (5). The node inconsistency of v_i is the average of local inconsistency between node v_i and all its neighbors $v_j \in N(v_i)$:

$$I_{node}(v_i) = \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} I_{local}(v_i, v_j). \tag{8}$$

We define the inconsistency loss as the average of total inconsistency over all nodes in the graph $\mathcal{G}$:

$$\mathcal{L}_{incon} = \frac{1}{N} \sum_{i=1}^{N} I_{node}(v_i) = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} I_{local}(v_i, v_j) \right]. \tag{9}$$

Note that Eq. (9) can be simplified as Eq. (10) when the local inconsistency $I_{local}(v_i, v_j)$ is defined as a symmetric function of node v_i and its neighbor v_j:

$$\mathcal{L}_{incon} = \frac{1}{N} \sum_{(i,j) \in \mathcal{E}} \left[\left(\frac{1}{|N(v_i)|} + \frac{1}{|N(v_j)|} \right) I_{local}(v_i, v_j) \right]. \tag{10}$$

Here, $\frac{1}{|N(v_i)|}$ is the weight for the local inconsistency $I_{local}(v_i, v_j)$ of node v_i and its neighbor v_j. The weight takes the maximum value when $N(v_i) = 1$, which means the node has only one neighbor. As the number of node neighbor increases, the weight becomes smaller. The inconsistency loss $\mathcal{L}_{incon}$ is actually a weighted sum of the local inconsistency between node pairs (v_i, v_j). The weight of $I_{local}(v_i, v_j)$ is proportional to $1/|N(v_i)|$. Therefore, nodes with fewer neighbors would appear less but contribute more in the inconsistency loss. The intuition behind the weights is that node with fewer neighbors is more likely to be the same class as its neighbor,thus taking more weights in the loss function. $\mathcal{L}_{incon}$ is unsupervised because no label information occurs in the definition. The inconsistency loss encourages nodes in the graph to match the predicted pmf of its neighbors' pmfs.

4.3 Combine supervised loss and inconsistency loss

For semi-supervised node classification, the objective function of GCN model is a supervised loss defined as the cross-entropy error between predicted values and the ground-truth labels,

$$\mathcal{L}_{sup} = -\sum_{l \in \mathcal{Y}_L} \sum_{c=1}^{C} Y_{lc} \log Z_{lc}. \tag{11}$$

The conventional supervised loss $\mathcal{L}_{sup}$ constraints the model to well fit the distribution of labeled nodes. However, the model can not generalize well on unlabeled nodes as samples are not independent identically distributed (i.i.d). To improve the generalization ability on testing instances, we add the inconsistency loss on the initial supervised loss. The new loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{sup} + k\mathcal{L}_{incon}, \tag{12}$$

where k is the tradeoff weight adjusting the proportion of inconsistency loss in total loss. Our proposed method is a more generalized form of GCN. When we set $k = 0$, the proposal is reduced to GCN.

Algorithm 1 describes the training process of our proposed PKGCN model. The loss function $\mathcal{L}$ is defined as the weighted sum of $\mathcal{L}_{sup}$ and $\mathcal{L}_{incon}$. In Algorithm 1, $k(t)$ is the tradeoff weight function. We propose two strategies to implement $k(t)$: static strategy and dynamic strategy. For static strategy, we choose a constant value of k with the highest classification accuracy on the validation set. For dynamic strategy, we first give $k(t)$ a initial value $k(t) = 0$ and then increase $k(t)$ as the number of training epoch increases. Our basic motivation is to train the model under more supervised information at the beginning and then transit to unsupervised information learning. We discuss different strategies for choosing the tradeoff weight function $k(t)$ in Sect. 5.4.2.

Algorithm 1 PKGCN

Input:
$\mathbf{X} \in \mathbb{R}^{N \times D}$: the data feature matrix
$\mathbf{Y} \in \mathbb{R}^{N \times C}$: the masked ground truth label matrix
$\mathcal{Y}_L$: the indices set of training data
$\mathbf{A}$: the adjacency matrix
$k(t)$: the tradeoff weight function
n_epoch: the number of training epochs
Output: The trained model
Randomly initialize $\mathbf{W}^{(0)}$ and $\mathbf{W}^{(1)}$ with Glorot initializer
for t in range(0, n_epoch) **do**
  $\mathbf{Z} \leftarrow \text{softmax}\left(\hat{\mathbf{A}}\ \text{ReLU}\left(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}\right)\mathbf{W}^{(1)}\right)$ ▷ Eq. (3)
  $\mathcal{L}_{sup} \leftarrow -\sum_{l \in \mathcal{Y}_L} \sum_{c=1}^{C} Y_{lc} \log Z_{lc}$ ▷ Eq. (11)
  $\mathcal{L}_{incon} \leftarrow \frac{1}{N} \sum_{i=1}^{N} I_{node}(v_i)$ ▷ Eq. (5) (8) (9)
  $\mathcal{L} \leftarrow \mathcal{L}_{sup} + k(t)\mathcal{L}_{incon}$ ▷ Eq. (12)
  Updating $\mathbf{W}^{(0)}$ and $\mathbf{W}^{(1)}$ using $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(0)}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}}$
end for

5 Experimental study

5.1 Datasets

For comparison, we utilize three standard citation network benchmark datasets: Cora, Citeseer, and Pubmed employed in previous study [19]. In the above three datasets, nodes are scientific publications and edges are citation links. We treat the links as undirected edges and construct a binary, symmetric adjacency matrix A. Zachary's karate club network [41] is used for case study in Section 5.4.3. Random

graphs are used for complexity analysis in 5.4.4. An overview of the statistics of the datasets is given in Table 1 and detailed descriptions are given below:

- *Cora* the Cora dataset contains 2708 scientific publications and 5429 citation links. Each document is described by a 0/1-valued word vector of 1433 dimensions and classified into one of 7 classes. Only 5.2% nodes are labeled for training.
- *Citeseer* the Citeseer dataset contains 3327 scientific publications and 4732 citation links. Each document is described by a 0/1-valued word vector of 3703 dimensions and classified into one of 6 classes. Only 3.6% nodes are labeled for training.
- *Pubmed* the Pubmed dataset contains 19717 scientific publications and 44,338 citation links. Each document is described by a Term-Frequency–Inverse Document Frequency (TF–IDF) vector of 500 dimensions and classified into one of 3 classes. Only 0.3% nodes are labeled for training.
- *Zachary's karate club* Zachary's karate club dataset is a well-known social network. The dataset contains 34 nodes and 77 edges. Each node represents a member of the club, and each edge represents that a relationship exists between two members. All members are classified into one of 2 groups after an argument between two teachers. We randomly choose 6 nodes for training.
- *Random graphs* we generate random graphs for experimental study of time complexity. For a graph with N nodes, we create $2N$ edges uniformly at random. For node v_i in the graph, we take *ith* row of the identity matrix I_N as its feature. For training set, we randomly sample 10% nodes as the labeled nodes and assign each node a random C-dimensional one-hot vector as its label. In our experiment, we set $C = 5$.

5.2 Baseline algorithms

We compare our proposed method PKGCN and its variants with the state-of-the-art model GCN [19] and other semi-supervised learning methods, including label propagation (LP) [44], semi-supervised embedding (SemiEmb) [38], manifold regularization (ManiReg) [2], Skip-gram based graph embeddings (DeepWalk) [33], iterative classification algorithm (ICA) [26] and Planetoid [39]. We also compare our model with ChebyNet in [8] using higher-order Chebyshev filters.

Table 1 Dataset statistics

Dataset	Nodes	Edges	Classes	Features	Label rate (%)
Cora	2708	5429	7	1433	5.2
Citeseer	3327	4732	6	3703	3.6
Pubmed	19717	44338	3	500	0.3
Zachary's karate club	34	77	2	34	17.6

5.3 Evaluation metric and experimental set-up

We strictly follow the training, validation and test data split provided by [19] and [39]. For all datasets, we use 20 nodes per class for training, 500 nodes for validation and 1000 nodes for test. Node labels in the validation set are not used for training. For the multi-classification problem, we evaluate the models with micro-f1 (accuracy) and macro-f1 on the test set.

For a fair comparison, all experiments in the paper are implemented with PyTorch 1.0.0 on GeForce® GTX 1080 Ti. Furthermore, we use the same hyper-parameters setting as in [19] except stopping rules: Adam [18] optimizer with a learning rate of 0.01, dropout rate of 0.5, L2 regularization weight of 5×10^{-4}, two convolutional layers, and 16 hidden units. We initialize weights using Glorot initialization [12] and row-normalize input feature vectors. We train the model on Cora, Citeseer and Pubmed for 1000, 1000, 200 epochs respectively.

The only hyper-parameter we need to choose is the tradeoff weight k in our model. In our experiment, we choose k with the highest accuracy on the validation set.

5.4 Results and analysis

In this subsection, we first compare the variants of PKGCN and the results of statistical test indicate PKGCN-EUC is the best model. Then, PKGCN is compared with GCN and other baselines and the results show that our proposed method is superior to GCN. We further analyze the effect of tradeoff between supervised loss and inconsistency loss and find that an appropriate setting of tradeoff is most effective. Finally, we show that the time complexity of PKGCN is in linear with the number of edges.

5.4.1 Semi-supervised node classification

First, we conduct experiments of GCN, PKGCN-JS and PKGCN-EUC with different training epochs. PKGCN-JS and PKGCN-EUC are variants of our proposed model using JS divergence and Euclidean distance as the distance measure function respectively. The results are shown in Tables 2, 3 and 4. We report the mean accuracy of 10 independent runs using different random seeds.

Two conclusions can be drawn from the above experimental results:

Table 2 Classification accuracies of GCN, PKGCN-JS and PKGCN-EUC on Cora dataset for different training epochs.

	GCN (%)	PKGCN-JS (%)	PKGCN-EUC (%)
200	81.25	82.05	82.36
400	80.89	82.45	83.16
600	80.81	82.63	83.45
800	80.96	82.83	83.82
1000	81.14	82.88	83.80
1200	80.93	82.94	83.68

Table 3 Classification accuracies of GCN, PKGCN-JS and PKGCN-EUC on Citeseer dataset for different training epochs

	GCN (%)	PKGCN-JS (%)	PKGCN-EUC (%)
200	70.05	70.46	70.48
400	69.62	70.63	71.09
600	69.74	70.88	71.24
800	69.87	70.81	71.16
1000	69.88	71.04	71.48
1200	69.43	70.61	71.33

Table 4 Classification accuracies of GCN, PKGCN-JS and PKGCN-EUC on Pubmed dataset for different training epochs

	GCN (%)	PKGCN-JS (%)	PKGCN-EUC (%)
200	78.07	78.16	78.47
400	77.83	77.89	78.00
600	77.81	77.71	77.71
800	77.60	77.66	77.75
1000	77.65	77.62	77.80
1200	77.64	77.44	77.65

- Under the same training epochs settings, both PKGCN-JS and PKGCN-EUC outperform GCN, and PKGCN-EUC outperforms PKGCN-JS;
- In general, the performance of GCN degrades as its training epoch increases (when exceeds 200). However, as the number of training epochs increases, the performance of PKGCN-JS and PKGCN-EUC increases.

Then, we run the experiments of variants of PKGCN with random weight initialization for 100 times. We report the mean micro-f1 and macro-f1 in Table 5. The result shows that PKGCN-EUC performs better than PKGCN-JS on Cora and Citeseer dataset considering both micro-f1 and macro-f1. On Pubmed dataset, PKGCN-JS has no difference in performance compared to PKGCN-JS. Then, we conduct the t-test to evaluate the significance of the results in Table 5. As shown in Table 6, the p values suggest that PKGCN-EUC is better than PKGCN-JS on Cora and Citeseer dataset under the significance level of $\alpha = 0.001$.

Table 5 Mean and standard deviations of micro-f1 and macro-f1 for PKGCN-JS and PKGCN-EUC

		PKGCN-JS	PKGCN-EUC
Cora	micro-f1	0.827 ± 0.0037 ($k = 6.0$)	$\mathbf{0.836 \pm 0.0038}$ ($k = 2.0$)
	macro-f1	0.817 ± 0.0033 ($k = 6.0$)	$\mathbf{0.824 \pm 0.0033}$ ($k = 2.0$)
Citeseer	micro-f1	0.708 ± 0.0057 ($k = 5.5$)	$\mathbf{0.717 \pm 0.0058}$ ($k = 1.5$)
	macro-f1	0.679 ± 0.0054 ($k = 5.5$)	$\mathbf{0.683 \pm 0.0053}$ ($k = 1.5$)
Pubmed	micro-f1	0.784 ± 0.0037 ($k = 2.0$)	0.784 ± 0.0040 ($k = 0.5$)
	macro-f1	0.777 ± 0.0035 ($k = 2.0$)	0.777 ± 0.0036 ($k = 0.5$)

The highest accuracy is highlighted in bold

Table 6 Results of t-test comparing PKGCN-JS and PKGCN-EUC using micro-f1 and macro-f1

Dataset	Metric	t	p value
Cora	micro-f1	16.9691	< 0.0010
	macro-f1	14.9992	< 0.0010
Citeseer	micro-f1	11.0673	< 0.0010
	macro-f1	5.2865	< 0.0010
Pubmed	micro-f1	0	0.5000
	macro-f1	0	0.5000

We report the mean accuracy of 100 runs with random weight initialization in Table 7. Results are summarized in Table 7 where the highest accuracy in each column is highlighted in bold. Results of baseline methods are taken from [19]. On Cora and Citeseer dataset, PKGCN-EUC outperforms the GCN model with an improvement of 2.1% and 1.4% in accuracy respectively. On Pubmed dataset, our methods perform comparably to GCN and outperform other baselines significantly.

To explain the results, we further calculate the inconsistency rate, defined as the proportion of edges linking two nodes belonging to different classes divided by the number of classes. As shown in Table 8, the inconsistency rate of Cora dataset is less than 2.71%, which verifies our observation statistically. Our model is based on the assumption that nearby nodes should have the same class labels and similar predicted pmf. The inconsistency rate can reflect how the

Table 7 Summary of results in terms of classification accuracies for Cora, Citeseer and Pubmed

Method	Cora (%)	Citeseer (%)	Pubmed (%)
ManiReg [2]	59.5	60.1	70.7
SemiEmb [38]	59.0	59.6	71.7
LP [44]	68.0	45.3	63.0
DeepWalk [33]	67.2	43.2	65.3
ICA [26]	75.1	69.1	73.9
Planetoid [39]	75.7	64.7	77.2
ChebyNet [8]	81.2	69.8	74.4
GCN [19]	81.5	70.3	**79.0**
PKGCN-JS (ours)	82.7 ($k = 6.0$)	70.8 ($k = 5.5$)	78.4 ($k = 2.0$)
PKGCN-EUC (ours)	**83.6** ($k = 2.0$)	**71.7** ($k = 1.5$)	78.4 ($k = 0.5$)

Table 8 The proportion of edges linking two nodes belonging to different classes and the inconsistency rate of Cora, Citeseer and Pubmed respectively

Dataset	Proportion (%)	Inconsistency rate (%)
Cora	18.97	2.71
Citeseer	26.10	4.35
Pubmed	19.80	6.60

graph data deviates from the assumption. Since the inconsistency rate of Pubmed dataset (6.60%) is higher than Cora (2.71%) and Citeseer (4.35%), the performance on Pubmed dataset is worse than that on Cora and Citeseer. This result indicates that our proposed PKGCN may not always be effective to improve GCN and the performance depends on the degree of the correspondence between the graph data and the assumption.

5.4.2 Effect of tradeoff between supervised loss and inconsistency loss

The tradeoff weight k is a critical parameter in our model, determining the proportion of inconsistency loss in the total loss. First, we analyze the effect of static tradeoff weight k on classification accuracy, supervised loss and inconsistency loss. In our experiment, we change k from 0 to 12 with a step of 1 every 100 epochs. For each k, we repeat the experiment 100 times. We plot the median curve, upper quartile and lower quartile on the test set in Fig. 2. When $k = 0$, the classification accuracy approximately equals to which of the GCN model. As k increases, the classification accuracy increases first and then decreases, however, the variance of accuracy decreases first and then increase (Fig. 2a, b). As k increases, more weight is placed on the inconsistency loss. Therefore the supervised loss increases (Fig. 2c, d) and the inconsistency loss decreases (Fig. 2e, f).

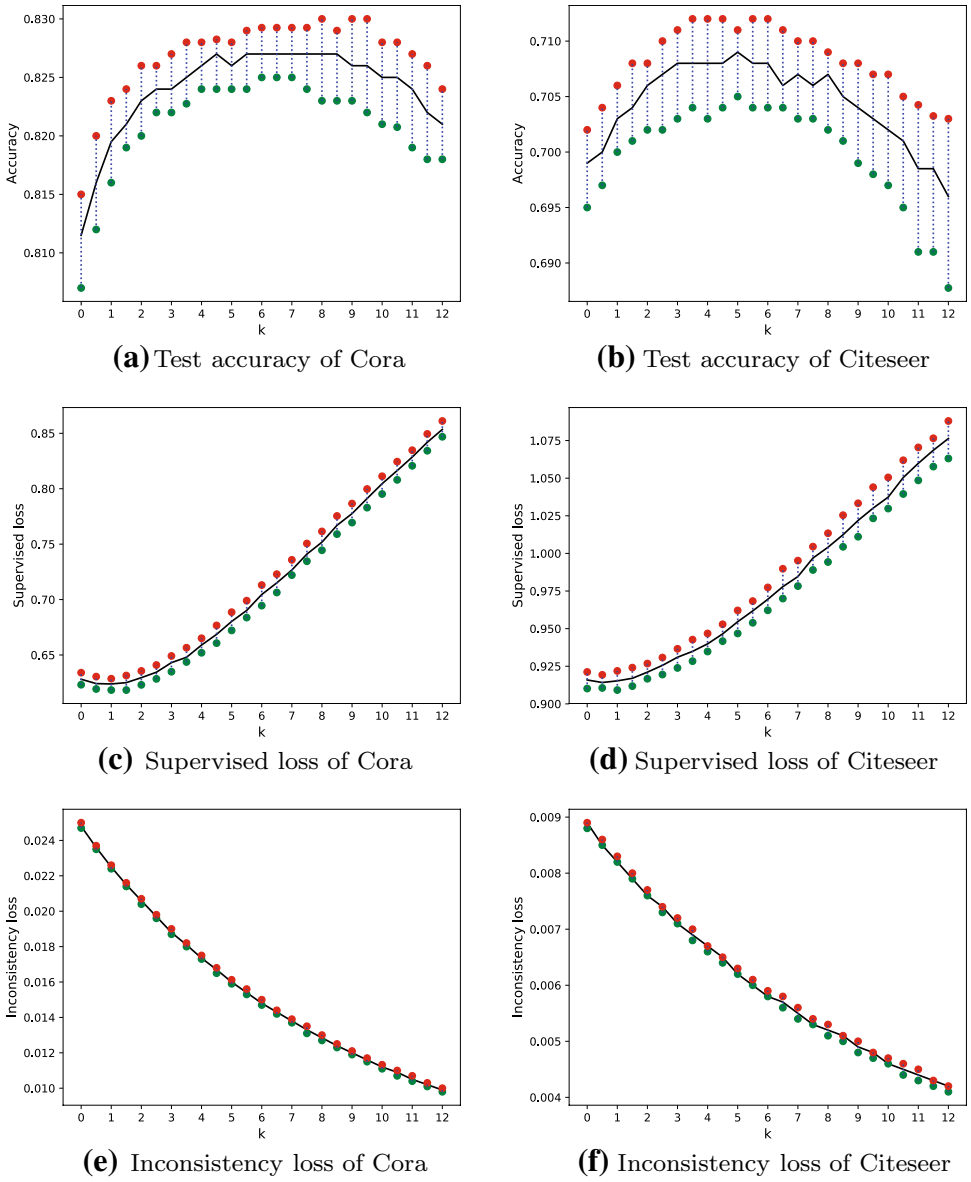
We then analyze the effect of dynamic tradeoff weight on classification accuracy. We devised several implementations of tradeoff weight function $k(t)$, see Tables 9, 10 and Fig. 3. Table 9 shows the classification accuracy of PKGCN-JS with different $k(t)$. All models with an increasing tradeoff weight function outperform the models with a decreasing one. Table 10 shows that PKGCN-EUC with an increasing tradeoff weight function outperforms PKGCN-EUC with a decreasing one. Furthermore, on Cora and Citeseer, PKGCN-EUC with a increasing dynamic tradeoff function (84.2% for Cora and 72.7% for Citeseer) outperforms the model with a fixed tradeoff in Tabel 7 (83.6% for Cora and 71.7% for Citeseer). The results verify our conjecture in Sect. 4.3. That is, the model needs the inconsistency loss in the late stage of the training process to improve generalization on unlabeled data.

5.4.3 Case study on Karate club network

We analyze why PKGCN is superior to GCN through visualizing the predicted results on the Karate club network. We set $k = 1$ for PKGCN-JS and run 10 epochs. Other parameters are the same as in Sect. 5.3. The results are shown in Fig. 4. Note that PKGCN correctly classified node 24 while GCN does not, as PKGCN imposes a penalty on the inconsistency of predicted class pmf between two adjacent nodes. The inconsistency loss encourages the model to predict the pmf of small-degree nodes to be consistent with their neighbors. Therefore, PKGCN can achieve better performance than GCN.

5.4.4 Computational complexity

As analysed in [19], the computational complexity of GCN is $\mathcal{O}(|\mathcal{E}|DHC)$, which is dominated by Eq. (3). Our proposed PKGCN is based on GCN with an additional inconsistency loss entry. Using the sparse matrix operations, the time complexity of evaluating Eq. (9) is $\mathcal{O}(|\mathcal{E}|C)$. Thus our proposed model has the same time complexity as the GCN model. We repeat the experiment on random graphs of different sizes for ten times and report the mean training time per epoch. See Sect. 5.1 for more details about random graph dataset used in our experiment. As shown in Fig. 5, the time complexity of PKGCN is linear in the number of graph edges.



(a) Test accuracy of Cora **(b)** Test accuracy of Citeseer

(c) Supervised loss of Cora **(d)** Supervised loss of Citeseer

(e) Inconsistency loss of Cora **(f)** Inconsistency loss of Citeseer

Fig. 2 The test classification accuracy (row 1), supervised loss (row 2) and inconsistency loss (row 3) of PKGCN-JS on the Citeseer (column 1) and Cora (column 2) datasets with tradeoff weight k changing from 0 to 12. Each experiment repeats 100 times. Black lines denote the median. Red and green dots denote the upper and lower quartiles respectively (color figure online)

Table 9 Classification accuracies of PKGCN-JS for Cora, Citeseer with different implementations of dynamic tradeoff weight function $k(t)$

		Cora (%)	Citeseer (%)
Increasing	$k_1(t) = t/100$	**82.8**	**70.7**
	$k_2(t) = t^2/10000$	82.5	70.1
Decreasing	$k_4(t) = 10 - t/100$	82.0	69.5
	$k_5(t) = 10 - t^2/10000$	81.4	69.3
	$k_6(t) = 10 - \sqrt{t/10}$	82.2	69.5

The highest accuracy is highlighted in bold

Table 10 Classification accuracies of PKGCN-EUC for Cora, Citeseer with different implementations of dynamic tradeoff weight function $k(t)$

$k(t)$		Cora (%)	Citeseer (%)
increasing	$k_1(t) = t/500$	84.1	**72.7**
	$k_2(t) = t^2/50000$	83.6	72.5
	$k_3(t) = \sqrt{t/250}$	**84.2**	72.4
decreasing	$k_4(t) = 2 - t/500$	81.6	70.2
	$k_5(t) = 2 - t^2/50000$	81.2	70.3
	$k_6(t) = 2 - \sqrt{t/250}$	82.3	70.7

The highest accuracy is highlighted in bold

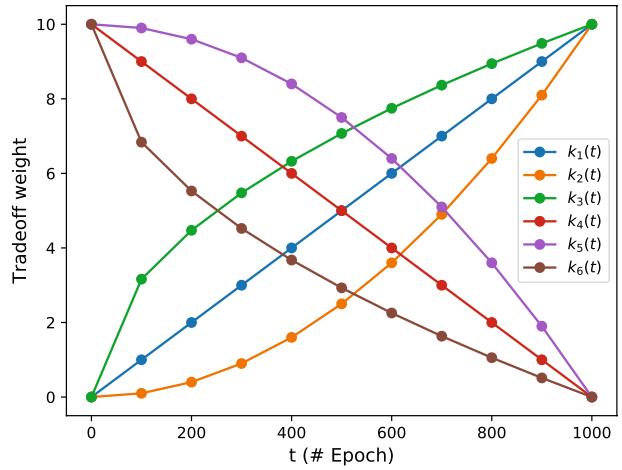


Fig. 3 Different implementations of dynamic tradeoff functions $k(t)$ in Algorithm 1. $k_1(t)$, $k_2(t)$ and $k_3(t)$ are increasing functions. $k_3(t)$, $k_4(t)$ and $k_(5)$ are decreasing functions. The value of each function changes from 0 to 10 (or from 10 to 0) with a step of $t = 100$. Details about the implement of functions can be found in Table 9

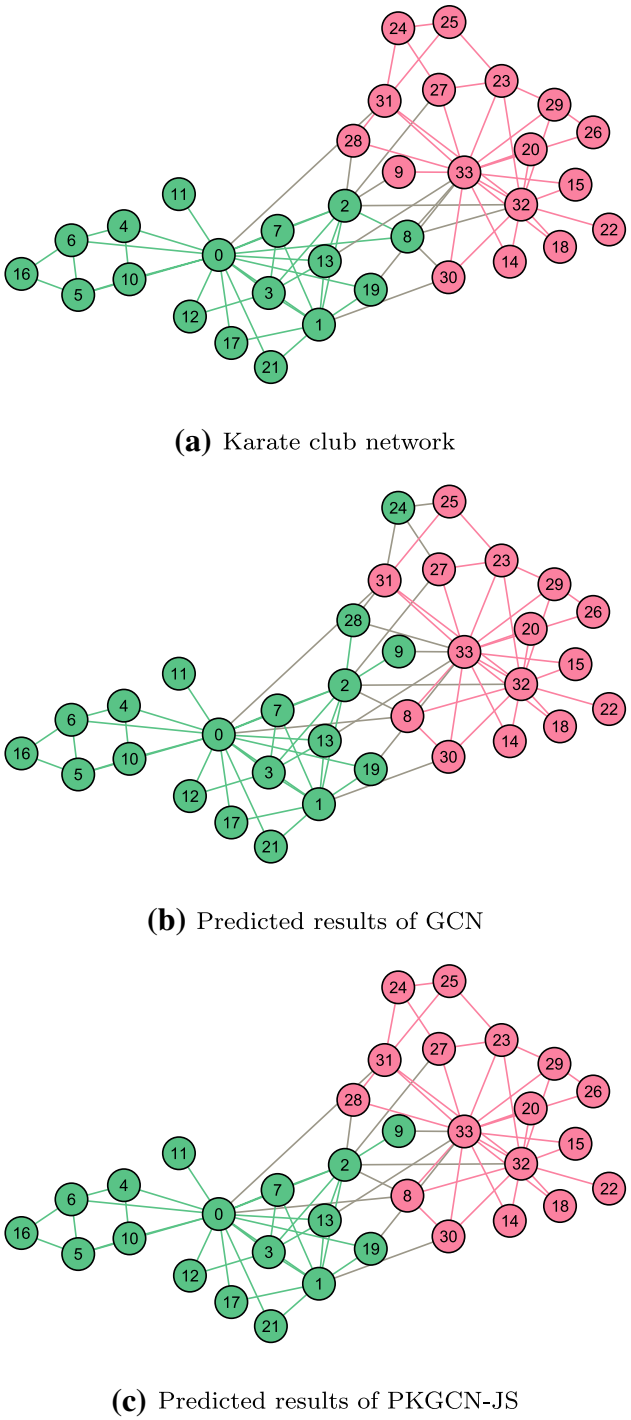
5.5 Discussion

5.5.1 Insights of PKGCN

Our proposed PKGCN enhances GCN through introducing the prior knowledge that adjacent nodes belong to the same class. The propagation rule and the network structure of PKGCN are the same with GCN. However its loss function includes an additional unsupervised term defined on both labeled and unlabeled nodes. With the prior knowledge, PKGCN achieves better generalization ability on unlabeled data. On datasets with low inconsistency rate such as Cora (2.71%) and Citeseer (4.35%), PKGCN improves GCN with 2.1% and 1.4% in accuracy respectively. This result indicates that prior knowledge contributes to improving GCN when the assumption is consistent with data. We further analyze the effect of tradeoff in our model. As shown in Fig. 2, the tradeoff parameter controls the importance of supervised term and unsupervised term in total loss. The relationship between accuracy and tradeoff is approximately quadratic and maximum is obtained with a medium value. Experiments about dynamic tradeoff indicate that PKGCN learns the supervised information in early stage, and generalizes to unlabeled data with prior knowledge in late stage. Finally, as shown in Fig. 5, PKGCN does not increase the time complexity when compared to GCN, thus our proposed method is effective and practical.

5.5.2 A general framework

Our proposed method is actually a special case under the Bayesian framework. We define the probability v_i and v_j connected by an edge having the same class label as $P(\text{same}, v_i, v_j)$. Using Bayesian formula, the probability can be decomposed into



(a) Karate club network

(b) Predicted results of GCN

(c) Predicted results of PKGCN-JS

Fig. 4 Visualization of the predicted results. **a** shows the Karate club network and the groundtruth label of each node. **b** presents the predicted results of GCN. **c** presents the predicted results of PKGCN-JS. Note that PKGCN-JS correctly classifies node 24 while GCN does not

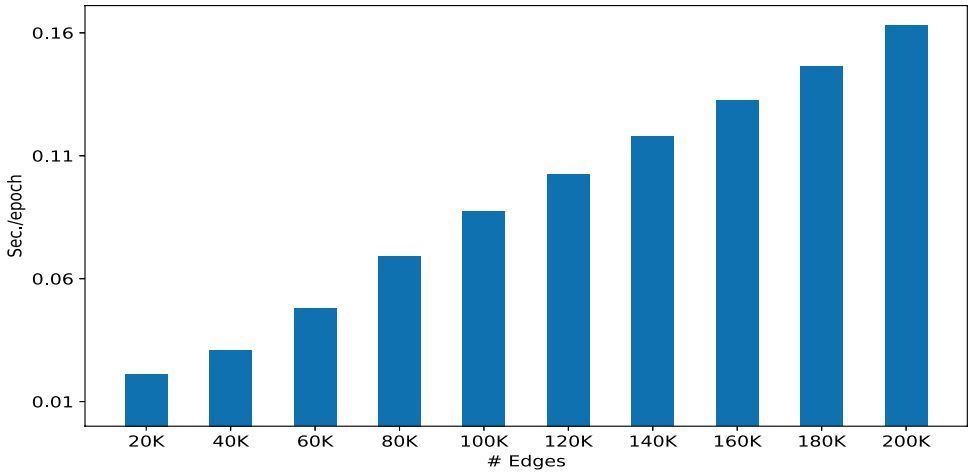


Fig. 5 Training time per epoch for random graphs

$$P(\text{same}, v_i, v_j) = P(v_i)P(v_j|v_i)P(\text{same}|v_i, v_j). \tag{13}$$

Here, $P(v_i)$ is the prior probability of node v_i. $P(v_j|v_i)$ is the probability of node v_j given node v_i. $P(\text{same}|v_i, v_j)$ is the probability that two nodes belong to the same class given v_i and v_j. The prior probabilities $P(v_i)$ and $P(v_j|v_i)$ are either given by an "oracle" or estimated from labeled data. In this paper, $P(v_i) = \frac{1}{N}$ and $P(v_j|v_i) = \frac{1}{|N(v_i)|}$. For $P(\text{same}|v_i, v_j)$, we define it to be inversely proportional to the local inconsistency as

$$P(\text{same}|v_i, v_j) = \frac{1}{1 + I_{local}(v_i, v_j)}. \tag{14}$$

$P(\text{same}|v_i, v_j)$ decreases from 1 to 0 as $I_{local}(v_i, v_j)$ increase from 0 to $+\infty$. Using the Taylor's theorem that $f(x) = \frac{1}{1+x} \approx 1 - x\,(x \approx 0)$, we can approximate Eq. (14) with

$$P(\text{same}|v_i, v_j) \approx 1 - I_{local}(v_i, v_j). \tag{15}$$

The objective is to maximize the sum of probability over all node pairs in graph,

$$\sum_{i=1}^{N} \sum_{j \in N(v_i)} P(\text{same}, v_i, v_j) = \text{const.} - \sum_{i=1}^{N} \sum_{j \in N(v_i)} \left[\frac{1}{N|N(v_i)|} I_{local}(v_i, v_j) \right], \tag{16}$$

which is equivalent to minimizing the inconsistency loss in Eq. (9). Under this framework, we can incorporate more prior knowledge in the model for using different prior probability $P(v_i)$ and $P(v_j|v_i)$, and adopt flexible definition of $P(\text{same}|v_i, v_j)$. For example, in the heterogeneous graph, we can define $P(\text{same}|v_i, v_j)$ to be related with the relation type between v_i and v_j, or the meta path from v_i to v_j.

5.5.3 Limitations and future work

Here, we list some limitations of our proposed method and point out some possible solutions.

First, since our model is based on GCN, the memory requirement of PKGCN is also linear in the size of dataset under the full-batch gradient descent optimization procedure. The recently proposed FastGCN [7] and GraphSAGE [15] overcome the limitation through sampling mini-batch nodes at each layer. An interesting research direction is applying our paradigm to sampling-based graph neural networks, where a new design of inconsistency loss would be needed.

Second, our model is currently limited to undirected, homogeneous and static graph. However, graph-structured data is usually directed, heterogeneous and dynamic in the real world. In addition, edge features are also important information in graph mining. In future work, we will investigate extending our model to varying types of graphs.

Third, the tradeoff parameter is determined through experiment on the validation set. However, we can use gradient descent to automatically learn a better parameter. In future, we will exploit how to ensemble tradeoff parameter optimization and model parameter learning together.

Fourth, the prior knowledge that adjacent nodes belong to the same category is commonly used as a basic assumption in graph-based semi-supervised learning [2, 42, 44]. The homophily theory has also been studied in sociology by [28]. However, the assumption is somewhat strong since it is valid only when the classes are well separated and clustered. One potential approach could be adopting a more general assumption that the labels of the k-hop neighbors ($k \geq 1$) may affect the node's category distribution and we can use attention mechanism (e.g. GAT [37]) to selectively capture these relations.

Finally, we will also study the method for quantitatively evaluating the consistency between prior knowledge and graph data. This helps us to understand what is the right time to use our model.

6 Conclusions

In this paper, we propose PKGCN which utilizes the prior knowledge on graphs to enhance GCN for graph-based semi-supervised learning. The prior knowledge that adjacent nodes have the same labels is incorporated through defining the optimization objective as a tradeoff between the supervised term and the unsupervised term. In our paradigm, the graph Laplacian-based models and GCN are combined to overcome the disadvantages of both methods. Furthermore, PKGCN does not increase time complexity while achieving a compelling performance. Comprehensive experiments show that PKGCN is better than GCN and significantly outperforms other models, which verified the effectiveness of our design encoding prior knowledge to the loss function.

Acknowledgements This work was funded by the National Natural Science Foundation of China under Grant No. U1636220 and 61532006, and Beijing Municipal Natural Science Foundation under Grant No. 4172063.

Compliance with ethical standards

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

References

1. Atwood J, Towsley DF (2016) Diffusion-convolutional neural networks. In: Advances in neural information processing systems, pp 1993–2001
2. Belkin M, Niyogi P, Sindhwani V, Bartlett P (2006) Manifold regularization: a geometric framework for learning from examples. J Mach Learn Res 7(1):2399–2434
3. Blum A, Mitchell TM (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th annual conference on computational learning theory, pp 92–100
4. Bronstein MM, Bruna J, Lecun Y, Szlam A, Vandergheynst P (2017) Geometric deep learning: going beyond euclidean data. IEEE Signal Process Mag 34(4):18–42
5. Bruna J, Zaremba W, Szlam A, Lecun Y (2014) Spectral networks and locally connected networks on graphs. In: Proceedings of the 2th international conference on learning representations
6. Chapelle OZA, Scholkopf B (2006) Semi-supervised learning. MIT Press, Cambridge
7. Chen J, Ma T, Xiao C (2018) Fastgcn: Fast learning with graph convolutional networks via importance sampling. In: Proceedings of the 6th international conference on learning representations
8. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. pp 3844–3852
9. Duvenaud DK, Maclaurin D, Aguileraiparraguirre J, Gomezbombarelli R, Hirzel TD, Aspuruguzik A, Adams RP (2015) Convolutional networks on graphs for learning molecular fingerprints. In: Advances in neural information processing systems, pp 2224–2232
10. Fout A, Byrd J, Shariat B, Benhur A (2017) Protein interface prediction using graph convolutional networks. In: Advances in neural information processing systems, pp 6530–6539
11. Fujino A, Ueda N, Saito K (2005) A hybrid generative/discriminative approach to semi-supervised classifier design. In: Proceedings of the 19th AAAI conference on artificial intelligence, pp 764–769
12. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th international conference on artificial intelligence and statistics, pp 249–256
13. Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: a survey. Knowl Based Syst 151:78–94
14. Grover A, Leskovec J (2016) Node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864
15. Hamilton WL, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in neural information processing systems, pp 1024–1034
16. Joachims T (1999) Transductive inference for text classification using support vector machines. In: Proceedings of the 16th international conference on machine learning, pp 200–209
17. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the conference on empirical methods in natural language processing, pp 1746–1751
18. Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In: Proceedings of the 3th international conference on learning representations
19. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th international conference on learning representations, pp 1–14
20. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
21. Kullback S, Leibler RA (1951) On information and sufficiency. Ann Math Stat 22(1):79–86
22. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86:2278–2324
23. Li Y, Tarlow D, Brockschmidt M, Zemel RS (2016) Gated graph sequence neural networks. In: Proceedings of the 5th international conference on learning representations
24. Li Y, Yu R, Shahabi C, Liu Y (2018) Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: Proceedings of the 27th international joint conference on artificial intelligence
25. Lin J (1991) Divergence measures based on the shannon entropy. IEEE Trans Inf Theory 37(1):145–151
26. Lu Q, Getoor L (2003) Link-based classification. In: Proceedings of the 20th international conference on machine learning, pp 496–503
27. Mallat S (1999) A wavelet tour of signal processing, 2nd edn. Elsevier, San Diego
28. Mcpherson M, Smithlovin L, Cook JM (2001) Birds of a feather: Homophily in social networks. Rev Soc 27(1):415–444
29. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: Proceedings of the neural information processing systems conference, pp 3111–3119
30. Monti F, Boscaini D, Masci J, Rodola E, Svoboda J, Bronstein MM (2017) Geometric deep learning on graphs and manifolds using mixture model cnns. pp 5425–5434
31. Niepert M, Ahmed MO, Kutzkov K (2016) Learning convolutional neural networks for graphs. In: Proceedings of the 20th international conference on machine learning, pp 2014–2023

32. Nigam K, Mccallum A, Thrun S, Mitchell TM (2000) Text classification from labeled and unlabeled documents using em. Mach Learn 39(2):103–134
33. Perozzi B, Al-Rfou R, Skiena S (2014) DeepWalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
34. Sainath TN, Vinyals O, Senior A, Sak H (2015) Convolutional, long short-term memory, fully connected deep neural networks. In: 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE, pp 4580–4584
35. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2009) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80
36. Shuman DI, Narang SK, 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 Process Mag 30(3):83–98
37. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2018) Graph attention networks. In: Proceedings of the 6th international conference on learning representations
38. Weston J, Ratle F, Mobahi H, Collobert R (2008) Deep learning via semi-supervised embedding. In: Proceedings of the 25th international conference on machine learning, pp 1168–1175
39. Yang Z, Cohen W, Salakhutdinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33th international conference on machine learning, pp 40–48
40. Yu B, Yin H, Zhu Z (2018) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. pp 3634–3640
41. Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33(4):452–473
42. Zhou D, Bousquet O, Lal TN, Weston J, Olkopf BS (2004) Learning with local and global consistency. Adv Neural Inf Process Syst 16:321–328
43. Zhu X (2005) Zhu X (2005) Semi-supervised learning literature survey. Tech. rep., University of Wisconsin-Madison Department of Computer Sciences
44. Zhu X, Ghahramani Z, Lafferty JD (2003) Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the 20th international conference on machine learning, pp 912–919