Cyclic label propagation for graph semi-supervised learning

Zhao Li[1] · Yixin Liu[2] · Zhen Zhang[3] · Shirui Pan[2] · Jianliang Gao[4] · Jiajun Bu[3]

Abstract

Graph neural networks (GNNs) have emerged as effective approaches for graph analysis, especially in the scenario of semi-supervised learning. Despite its success, GNN often suffers from over-smoothing and over-fitting problems, which affects its performance on node classification tasks. We analyze that an alternative method, the label propagation algorithm (LPA), avoids the aforementioned problems thus it is a promising choice for graph semi-supervised learning. Nevertheless, the intrinsic limitations of LPA on feature exploitation and relation modeling make propagating labels become less effective. To overcome these limitations, we introduce a novel framework for graph semi-supervised learning termed as **Cyc**lic Label **Prop**agation (**CycProp** for abbreviation), which integrates GNNs into the process of label propagation in a cyclic and mutually reinforcing manner to exploit the advantages of both GNNs and LPA. In particular, our proposed CycProp updates the node embeddings learned by GNN module with the augmented information by label propagation, while fine-tunes the weighted graph of label propagation with the help of node embedding in turn. After the model converges, reliably predicted labels and informative node embeddings are obtained with the LPA and GNN modules respectively. Extensive experiments on various real-world datasets are conducted, and the experimental results empirically demonstrate that the proposed CycProp model can achieve relatively significant gains over the state-of-the-art methods.

This article belongs to the Topical Collection: *Special Issue on Large Scale Graph Data Analytics*
Guest Editors: Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang

---

✉ Zhao Li
lizhao.lz@alibaba-inc.com

1   Alibaba Group, Hangzhou, China

2   Monash University, Clayton, Australia

3   Zhejiang University, Hangzhou, China

4   Central South University, Changsha, China

1 Introduction

Graph-structured data is pervasive in various applications, ranging from citation networks, social networks to E-commerce networks. Mining knowledge in graphs such as predicting node properties is desirable and meaningful to both academic and industrial communities. For example, given an academic citation network, we may be interested in predicting the research area of an author. Making such predictions has become the focus of graph analysis which broadly includes graph classification [13], link prediction [21] and community detection [16], *etc*. Among various graph analysis problems, semi-supervised node classification for graphs is an essential and widespread task, and it has attracted great interests [12, 35, 46].

Graph representation learning is an effective technique for tackling this task. Early shallow approaches [7, 29] typically follow a two-step framework, which aims to learn a continuous, compact, and low-dimensional embedding (vector) for each node in the graph. These embeddings are further fed into a classifier to infer the labels of nodes. Since the node representations are not optimized for the specific classifier, this two-step process will inevitably lead to sub-optimal performance. More recently, several semi-supervised graph neural networks (GNNs) [12, 35, 41] were proposed. They utilize deep learning techniques [15] such as convolution or attention mechanism to encode both the local graph structure and node attributes to generate embeddings, which are then followed by a prediction layer (e.g., softmax or logistic sigmoid function) for the classification purpose. Due to the powerful feature extraction ability of deep learning and the integrated end-to-end framework, they have achieved state-of-the-art performance in the node classification task.

While these GNNs approaches have become the de facto solution for graph semi-supervised learning, they still suffer from two shortcomings, the *over-smoothing* and *over-fitting* problems, due to the inherent training and test procedure for semi-supervised learning [30]. First, GNN essentially employs a message passing neural network with neighborhood representation aggregation to train a model and map the feature space into the label space [6, 47]. When the network architecture goes deep, due to the excessive aggregation, all nodes' representations tend to converge to a stationary point, resulting in the indistinguishable representations of nodes in different classes [4]. Such an issue is denoted as *over-smoothing* problem which seriously affects the performance of GNNs [18]. Second, GNN aims to minimize a loss function over the labeled nodes, which are typically very limited in semi-supervised learning. Therefore, GNNs easily *overfit* the training samples, leading to a degenerated generalization performance.

In semi-supervised learning, one alternative and promising learning paradigm is the label propagation algorithm (LPA) [48, 49]. Different from representation learning or GNNs, LPA builds up a graph over the labeled and unlabeled data, where edges in the graph connect semantically similar nodes with its weights reflecting how strong the similarities are. Then, LPA infers the labels of unlabeled nodes by propagating known labels through neighbors of each node. The edge weights in LPA are often set heuristically based on the observed node attributes (e.g., using Gaussian kernel function).

LPA has nice properties that can avoid both the *over-smoothing* and *over-fitting* problems faced by GNNs. Concretely, GNNs learn the feature mapping with multiple layers of aggregation, which would cause *over-smoothing* (node embeddings become indistinguishable) with the networks go deep. In contrast, LPA directly spreads the labels on a graph without involving the feature learning process. As a result, LPA would not lead to indistinguishable

node representations. Furthermore, LPA classifies nodes by propagating the labels instead of training a classifier to fit the limited training data. Therefore, the learned model would not *over-fit* the patterns in training set.

However, propagating labels effectively is not trivial, since the classic LPA still has the following intrinsic limitations.

– *Limitation 1: Limited capacity to exploit features.* Classic LPA derives edge weights from original high dimensional node attribute space where a large portion of sparse, redundant or noisy information is contained. These approaches cannot exploit the expressiveness of features effectively. Furthermore, computing the similarity of the raw attribute directly may lead to noisy weight values and the loss of key information.
– *Limitation 2: Hardly capture the strength of relation corresponding to the labels.* The edge weight is computed as a separate step from the label propagation in LPA. As a result, the label information is ignored when capturing the strength of the relation. As the edge weights are only calculated once based on the similarity of raw attributes, they cannot be updated reversely by the label propagation process. The fixed edge weights limit the classification performance for semi-supervised classification.

To overcome the aforementioned limitations, in this paper, we propose a novel framework for graph semi-supervised learning named **Cyc**lic Label **Prop**agation (**CycProp** for abbreviation). Our theme is to integrate GNNs into the process of label propagation in a cyclic and mutually reinforcing manner to exploit the advantages of both GNNs and LPA algorithms. More specifically, to overcome **Limitation 1**, we derive a novel label-adaptive graph neural network module to learn low-dimensional embeddings of nodes in a graph. To enhance the representation power of the embedding, we exploit the highly reliable labels obtained from label propagation in the negative sampling process, so that the label information can be nicely injected into the node embedding component. For **Limitation 2**, we develop an embedding-adaptive label propagation module, which utilizes the node embeddings to refine the edge weights for label propagation. With the label information injected in the node embeddings, the weights essentially capture the strength of the relations corresponding to the labels. A self-paced learning manner is devised to adaptively control the cyclic learning process, where the embedding learning and edge refining are optimized alternatively to mutually benefit each other in our framework. Once the model has converged, the unknown node labels can be obtained on-the-fly without training an extra classifier or performing a sophisticated inference procedure. A concept map of our framework is given in Figure 1.

To summarize, the main contributions are as follows:

– We propose CycProp, a unified graph semi-supervised learning framework that exploits the advantages of both GNN and LPA. Our novel presented framework updating node representation and weighted graph in a cyclic and mutually reinforcing way, our proposed framework can obtain label estimations and node embeddings simultaneously.
– We design a novel label-adaptive graph neural network module for graph representation learning, which leverages not only structure context but also self-adaptive augmented label context to learn the node embeddings.
– We conduct extensive experiments on various datasets to demonstrate the effectiveness of CycProp and its superiority compared to a range of state-of-the-art methods.
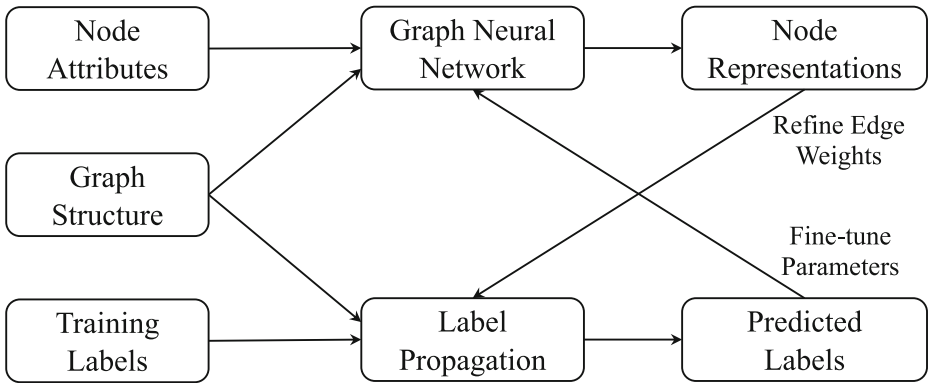
Figure 1 A concept map of our proposed framework. The node representations and the predicted labels update each other in a cyclic manner, which benefits each of the components to have a better performance

2 Related work

2.1 Graph representation learning

Graph embedding, an important branch of graph representation learning aims to embed nodes into latent vector spaces, where the inherent properties of the graph are preserved. Motivated by the success of Word2vec [25, 26], Skip-gram model is employed and adapted from word embeddings to node embeddings based on the graph topology. For instance, DeepWalk [29] and node2vec [7] use different sampling strategies to generate random walk sequences, which are then fed into Skip-gram model to learn low-dimensional embedding vectors. LINE [33] optimizes both the first-order and second-order proximity preserving objectives. While the above methods only utilize the graph structure information, some recent approaches attempts to consider preserving both the structure and attribute proximities in a unified space. For example, SNE [20] leverages a deep neural network architecture to capture the complex interrelations between graph structure and node attributes information. GraphSAGE [8] generates embeddings by sampling and aggregating attributes from nodes' local neighborhoods in an inductive setting. EP [5] tries to learn vector representations by utilizing a propagation design.

Another category of graph embedding algorithms follows a semi-supervised manner, in which available information includes not only node attributes but also node labels. Among them, TriDNR [27] models tri-party information sources including node structure, node attributes and node labels to jointly learn node representations. Planetoid [46] simultaneously optimizes the prediction of known labels and its corresponding graph contexts to learn node's representations. SEANO [19] learns low-dimensional node representation that considers the topological proximity, attribute affinity and label similarity. MDAL [42] introduces a multi-task dual attention LSTM model to capture multiple information for graph semi-supervised learning.

Recently, graph neural networks (GNNs) have raised extensive attention and attained state-of-the-art performance in several graph analysis applications, especially in semi-supervised node classification task. By applying deep learning techniques [15] to non-Euclidean domains, GNNs can learn node representation from high-dimensional feature

space and predicted labels simultaneously in an end-to-end way [43]. GCN [12], a representative model of GNNs, performs spectral convolutions on graph to encode both local graph structure and attributes of nodes into hidden representations. GAT [35] applies attention mechanism to attend over node's neighborhood contents for generating node embeddings. SGC [41] simplifies GCN by removing the non-linearity and also achieves competitive performance. GIN [45] borrows the power of graph isomorphism test to GNNs's design, and DGI [36] leverages a contrastive manner to train GNNs in an unsupervise way. Beyond the graph domain, GNNs are also applied to solving various machine learning problems, including time-series prediction [44], object detection [31] and few-shot learning [22, 23]. Unfortunately, GNNs for node classification usually suffer from two main obstacles, *over-fitting* and *over-smoothing*, which seriously hurt the performance of models [30].

In this paper, we leverage GNNs as an important component in our learning framework, but successfully avoid the obstacles of GNNs with the help of label propagation. Compared to the aforementioned methods, CycProp would not suffer from *over-fitting* and *over-smoothing* when making prediction for classification.

2.2 Label propagation algorithm

Label propagation algorithm (LPA) has been proposed as an efficient method to learn missing labels for graph data in a semi-supervised setting. GFHF [49] learns the predicted labels by optimizing the harmonic functions based on a Gaussian random field model. LLGC [48] considers local and global prior consistency through combining a smoothness constraint and a fitting constraint. LNP [38] studies the graph construction by approximating the whole graph with linear neighborhood structures, where labels are propagated to the remaining unlabeled nodes using the constructed graph. DLP [37] deals with multi-label propagation problem via considering the label correlation information. Moreover, inspired by LPA's formulation, other common approaches try to train a supervised learner to classify data features while regularizing it using graph information. For example, manifold regularization [1] trains a support vector machine with a graph Laplacian regularizer. LSHM [11] addresses the node classification task in heterogeneous social networks via the learned node representations.

Several recent works also exploit LPA to neural networks. For instance, NGM [3] utilizes the power of neural networks and constrains neighborhood nodes to learn similar representations for classification. LP-DSSL [10] utilizes label propagation to generate pseudo-labels for the unlabeled data, which expands the training sample set for neural network training. GCN-LPA [39] builds a GCN with learnable edge weights and views LPA as regularization to assist the GCN in learning proper edge weights. Our proposed CycProp also combines label propagation and neural network, but has several essential differences with the above methods: (1) All of the aforementioned methods predict the unknown label by neural networks, whereas CycProp makes classification prediction by label propagation, which effectively avoids *over-smoothing* and *over-fitting* problems; (2) In the above methods, the main component of objective functions is the cross-entropy loss function and LPA serves as a regularization term or pseudo-label generator. In contrast, we set the label propagation loss as the main objective and also design a structure-label-aware graph embedding loss function.

Compared to the traditional LPA methods, CycProp introduces GNNs to revise the edge weights iteratively. By such an adaptive weighting propagation, it has a significant advantage on classification performance.

3 Cyclic label propagation for graph semi-supervised learning

In this section, we first define the notations and present our problem formulation. Then, we introduce the two major components in our unified framework: (1) label-adaptive graph neural network module, and (2) embedding-adaptive label propagation module. After that, a joint training framework that integrates the two components is presented.

3.1 Notations and problem formulation

Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V} = \{v_1, \cdots, v_l, v_{l+1}, \cdots, v_n\}$ and $\mathcal{E}$ denote the set of nodes and edges, respectively; $\mathbf{X} \in \mathbb{R}^{n \times m}$ is a matrix that represents all node attributes, and $\mathbf{x}_i \in \mathbb{R}^m$ denotes the attributes affiliated with node v_i. Let label set $\{1, 2, \cdots, K\}$ represent different classes of labels and $\mathbf{Y} \in \mathbb{R}^{l \times K}$ be a label matrix, in which $\mathbf{y}_i \in \mathbb{R}^K$ denotes the label distribution of node v_i, i.e., if v_i belongs to class j, then $y_{ij} = 1$, otherwise $y_{ij} = 0$. The first l nodes v_i ($1 \leq i \leq l$) are labeled, and the remaining nodes v_u ($l+1 \leq u \leq n$) are unlabeled. With the above notations, we formally define our problem as follows.

Definition 1 Given an attributed graph $\mathcal{G}$, with partially labeled nodes $\{v_1, \ldots, v_l\}$ and the desired node embedding dimension d, our goal is to learn the label assignments $\mathbf{F} \in \mathbb{R}^{n \times K}$ and node embeddings $\mathbf{E} \in \mathbb{R}^{n \times d}$ simultaneously. Each node has a probability distribution over the set of labels.

3.2 Label-adaptive graph neural network module

To learn meaningful node embeddings, it's desirable to incorporate various available graph information. Different from [32] that models the attribute information as augmented nodes or [9] that uses node attributes to calculate a similarity matrix, we designed a label-adaptive graph neural network module to capture node's deep semantics.

Specifically, we generate node embeddings as follows,

$$\mathbf{e}_i = g_{\boldsymbol{\theta}}(\mathbf{x}_i), \tag{1}$$

where $g_{\boldsymbol{\theta}}(\cdot)$ can be any kind of GNNs [12, 35, 41, 43], and $\boldsymbol{\theta}$ is the parameter set. These methods typically work by propagating representations throughout the graph. Here, we choose GraphSAGE [8] as our graph neural network module in the experiments, due to its effectiveness and efficiency. Then, we optimize it in a label-adaptive manner by predicting its associated graph context. Formally, let (i, c) represent the node-context pair, i.e., node v_c is the graph context of node v_i, and our goal is to minimize the following log softmax probability,

$$-\log \sigma(\mathbf{e}_c^{\mathrm{T}} \mathbf{e}_i) - \sum_{s=1}^{s_{neg}} \mathbb{E}_{v_j \sim P_n(v)}[\log \sigma(-\mathbf{e}_j^{\mathrm{T}} \mathbf{e}_i)], \tag{2}$$

where $P_n(v) \propto d_v^{3/4}$ as suggested in [26], and d_v is the degree of node v. Then, the goal is indeed transformed to classify the node-context pairs (i, c) into positive context ($\gamma = +1$) or negative context ($\gamma = -1$) sampled from a noisy distribution. Therefore, the graph embedding loss with negative sampling can be rewritten as,

$$\mathcal{L}_{GE} = -\sum_{i=1}^{n} \mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{e}_c^{\mathrm{T}} \mathbf{e}_i), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function, i.e., $\sigma(x) = 1/(1 + e^{-x})$.

Algorithm 1 Structure-label-aware graph context sampling.

Require: Graph $\mathcal{G}$, indicator vector $\boldsymbol{\varphi}$, negative sample number s_{neg}, probability r, noise distribution $P_n(v)$.

Ensure: Graph context pairs (i, c, γ).

1: Draw a number $rnd \sim Uniform(0, 1)$.
2: **if** $rnd < r$ **then**
3:      // *sampling structure context*
4:      Uniformly sample a positive pair $(i, c, +1)$ with v_c from node v_i's neighborhood.
5:      Sample s_{neg} negative pairs $(i, c, -1)$ with c from $\mathcal{V}$ corresponding to $P_n(c)$.
6: **else**
7:      // *sampling label context*
8:      Uniformly sample a node v_i from nonzero elements in $\boldsymbol{\varphi}$.
9:      Uniformly sample a positive pair $(i, c, +1)$ with $\varphi_c = 1$ and $y_i = y_c$.
10:      Sample s_{neg} negative pair $(i, c, -1)$ with c from $\mathcal{V}$ with $\varphi_c = 1$ and $y_i \neq y_c$ corresponding to $P_n(c)$.
11: **end if**

We now present how to generate (i, c, γ) graph context pairs with a structure-label-aware sampling process. We develop two graph context sampling mechanisms, which is depicted in Algorithm 1.

Structure aware graph context sampling The first type is based on the graph structure, which encodes the structure information and regards the neighborhood nodes as positive node-context pairs.

Label aware graph context sampling The second type is based on the label set, which injects label information into the context and treats the nodes having the same labels as positive node-context pairs. Moreover, an indicator variable of $\boldsymbol{\varphi}$ is introduced to control the label related graph context. We first initialize the 0/1 indicator vector $\boldsymbol{\varphi} \in \mathbb{R}^n$ with the known label set, i.e., $\varphi_i = 1$ means node v_i is a label context candidate. Then we augment it through the label propagation module where the learned highly reliable labeled nodes will be gradually incorporated into $\boldsymbol{\varphi}$ to expand the label context candidates. The generated context pairs will be dynamically refreshed during the training process, due to the updating of parameter $\boldsymbol{\varphi}$ and node labels. The details about how to model the indicator vector $\boldsymbol{\varphi}$ will be described in the following subsection.

3.3 Embedding-adaptive label propagation module

In order to overcome the limitations of existing label propagation approaches [37, 38, 48], we propose to infer the edge weights in an informative embedding space via a mutually reinforcing manner. In detail, for each node v_i, its corresponding embedding vector $\mathbf{e}_i \in \mathbb{R}^d$ is obtained from the label-adaptive graph neural network module where d is the embedding dimension ($d \ll |m|$). Basically, edge weights can be calculated based on the following function similarly to [38, 48, 49],

$$s_{ij} = \exp\left(-\frac{\|\mathbf{e}_i - \mathbf{e}_j\|^2}{2\delta^2}\right), \tag{4}$$

where δ is the length scale parameter. With this measure, the estimated edge weights reflect the degree of similarities between each connected node pair, which will be dynamically adjusted in the training procedure as the updating of node embeddings.

The key to semi-supervised learning on graphs is to be in line with the prior consistency, such that the label is smooth over the graph. Following this principle, we devise the regularized objective function for embedding-adaptive label propagation as follows,

$$
\begin{aligned}
\mathcal{L}_{LP} = &\sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} s_{ij} \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 + \mu \sum_{i=1}^{l} \|\mathbf{f}_i - \mathbf{y}_i\|_2^2 \\
&+ \sum_{i=1}^{n} \varphi_i H(\mathbf{f}_i) + \lambda \sum_{i=1}^{n} -\varphi_i
\end{aligned}
\tag{5}
$$

$$
\text{s.t. } f_{ik} \geq 0; \ \sum_{k=1}^{K} f_{ik} = 1; \ \varphi_i \in \{0, 1\}, \ i = \{1, \cdots, n\},
$$

where s_{ij} is the edge weight between node v_i and v_j calculated according to (4). $\mathbf{y}_i$ indicates the ground truth label and $\mathcal{N}(i)$ represents the neighborhood of node v_i. μ is a trade-off hyper parameter between the smoothness and fitness terms, $\mathbf{f}_i$ is the learned label distribution of node v_i. Through this manner, the label propagation procedure benefits from the graph neural network module.

In addition, we introduce a self-paced regularizer [14] to prioritize label learning task as well as selecting some highly reliable node labels in each training iteration. The regularizer is composed of a Shannon entropy function $H(\cdot)$ and an indicator variable $\boldsymbol{\varphi}$. $H(\mathbf{f}_i)$ prevents the uniform label distribution, and is formally defined as follows,

$$
H(\mathbf{f}_i) = -\sum_{k=1}^{K} f_{ik} \times \log(f_{ik}),
\tag{6}
$$

where f_{ik} denotes the probability of node v_i belonging to class k. The smaller the Shannon entropy is, the larger the amount of information it contains. Specifically, a small Shannon entropy implies that $\mathbf{f}_i$ has a significantly higher probability value in one specific class. For instance, in an extreme case, if the probability of node v_i in class k is 1, the Shannon entropy of $\mathbf{f}_i$ will be 0. The self-paced parameter $\boldsymbol{\varphi}$ serves as an indicator vector in graph neural network module, which determines the potential label context based on its current label information. The binary value of φ_i indicates whether node v_i's learned label is reliable or not and λ acts as a threshold to distinguish the informative labels from the uninformative labels. If the Shannon entropy of $\mathbf{f}_i$ is smaller than the threshold, we set φ_i as 1 to indicate that node v_i can be utilized as a label context. As the training process goes on, λ is gradually increased such that more learned highly reliable labels can be included in graph embedding procedure to adaptively update node embeddings. In this way, the graph embedding procedure benefits from the label propagation module. Thus, the overall framework naturally forms a closed-loop via a mutually reinforcing manner.

3.4 CycProp: a joint learning framework

As is demonstrated before, each of the two modules can learn beneficial information from the other. Hence, we design a cyclic learning framework where the two modules are trained in an iterative and alternate way. Figure 2 provides an overview of our proposed CycProp
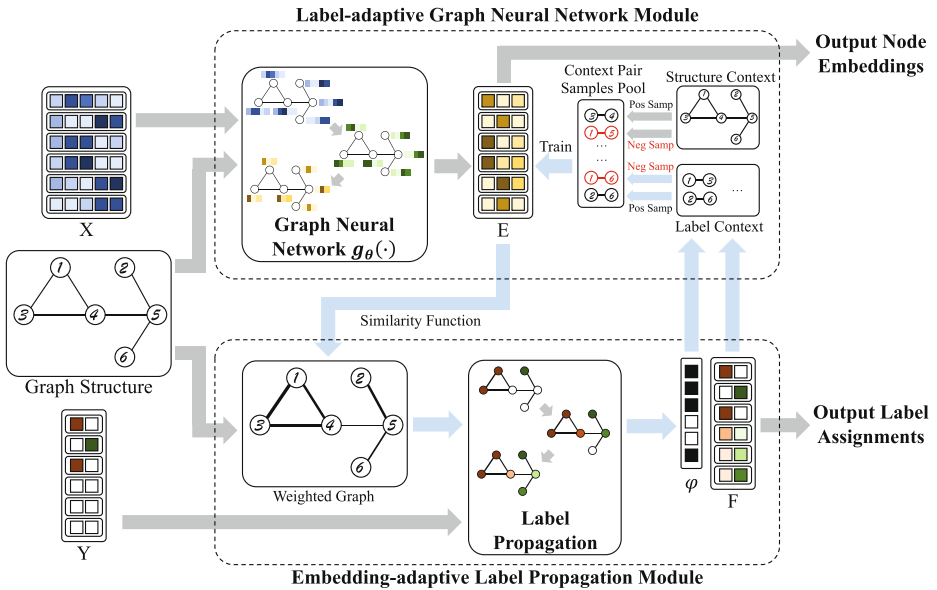
Figure 2 The overall framework of the CycProp model. The nodes 1-3 have known labels while the labels of nodes 4-6 need to be predicted. In the weighted graph, the thicker edges denote larger weights. In the context pair samples pool, the context pairs in black denote the positive samples ($\gamma = 1$), and which in red denote the negative samples ($\gamma = -1$). The arrows in blue indicate the data flow of cyclic learning

model. The two main components of our proposed framework, label-adaptive graph neural network module and embedding-adaptive label propagation module, output the node embeddings and label assignments (classification results) respectively. Firstly, in the label-adaptive graph neural network, the attribute matrix **X** and graph structure information are fed into the graph neural network $g_\theta(\cdot)$, which generates the node embeddings **E**. Then, in the embedding-adaptive label propagation module, the weighted graph is computed by a similarity function with the node embeddings **E**. After that, the predicted label assignments **F** and the indicator variable φ are obtained by label propagation with the weighted graph and known labels **Y**. Next, the predicted label and indicator are exploited to generate the label context, which is used to train the graph neural network in turn together with the structural context. In this way, the node embeddings **E** is updated to generate a more reasonable weight graph, which further optimizes the prediction **F** of label propagation. As the blue arrows are shown, by joint learning **E** and **F** in such a cyclic and mutually beneficial manner, finally, the model can output informative node embeddings as well as accurate label predictions. It is worth noticing that the final classification results are acquired by the label propagation module, which efficiently prevents the *over-smoothing* and *over-fitting* problems caused by GNNs.

The objective function of the proposed model is formulated as the weighted combination of $\mathcal{L}_{LP}$ and $\mathcal{L}_{GE}$ defined in Equations (3) and (5),

$$\mathcal{L} = \mathcal{L}_{LP} + \alpha \mathcal{L}_{GE}. \tag{7}$$

Considering the related parameters of both terms in $\mathcal{L}$, the set of learnable parameters in CycProp model is denoted as $\{\mathbf{F}, \varphi, \theta\}$. To minimize $\mathcal{L}$, we propose to employ stochastic

gradient descent [2] and proximal algorithm [28] to optimize our model via an alternative updating manner. We first give the partial derivative details of some key parameters as follows.

Updating F We utilize proximal gradient descent [17, 28] to solve this constrained optimization problem. In proximal algorithms, the interface to meet the constraint terms is via the proximal operator. Satisfying the non-negative and sum-to-1 constraints in our objective function, i.e., $\mathcal{D} = \{\mathbf{f}|\mathbf{f} \geq 0, \mathbf{f}^{\mathrm{T}}\mathbf{1} = 1\}$, is known as computing the projection onto the probability simplex. Here, we employ an efficient algorithm proposed in [40] to calculate the proximal operator. The procedure of finding $\mathbf{f} \in \mathcal{D}$ given $\mathbf{z}$ is shown in Algorithm 2.

Algorithm 2 Projection onto the probability simplex.

Require: $\mathbf{z} \in \mathbb{R}^K$.
 1: Sort $\mathbf{z}$ into $\mathbf{u}$: $u_1 \geq u_2 \geq \cdots \geq u_K$.
 2: Find $\rho = \max\{1 \leq j \leq K : u_j + \frac{1}{j}(1 - \sum_{i=1}^{j} u_i) > 0\}$.
 3: Define $\eta = \frac{1}{\rho}(1 - \sum_{i=1}^{\rho} u_i)$.
Ensure: $\mathbf{f}$, s.t. $f_i = \max\{z_i + \eta, 0\}, i = 1, 2, \cdots, K$.

Then we have the following equation as our proximal operator,

$$\mathbf{f} = \mathbf{prox}_{\mathcal{D}}(\mathbf{z}) = (\mathbf{z} + \eta\mathbf{1})_+, \tag{8}$$

where $(x)_+ = \max\{x, 0\}$ and η is computed according to the procedure shown in Algorithm 2. Note that the proximal operator always keeps our updated label distribution satisfied with the constraint $\mathcal{D}$. The partial derivative of $\mathbf{f}_i$ can be formulated as,

$$\frac{\partial \mathcal{L}}{\mathbf{f}_i} = \sum_{j \in \mathcal{N}(i)} 2s_{ij}(\mathbf{f}_i - \mathbf{f}_j) + 2\mu(\mathbf{f}_i - \mathbf{y}_i) \cdot \mathbb{I}(i) - \varphi_i(\log(\mathbf{f}_i) + \mathbf{1}), \tag{9}$$

where $\mathbb{I}(i)$ is an indicator function to indicate whether i is a labeled node or not and $\mathbf{1} \in \mathbb{R}^K$ represents the all one vector. Based on the proximal gradient method, parameter $\mathbf{f}_i$ can be updated as following,

$$\mathbf{f}_i = \mathbf{prox}_{\mathcal{D}}(\mathbf{f}_i - lr \cdot \frac{\partial \mathcal{L}}{\mathbf{f}_i}), \tag{10}$$

where lr denotes the learning rate in gradient descent.

Updating φ We first relax φ to take any real value in the interval $[0, 1]$. Then the partial derivative of φ_i can be derived as,

$$\frac{\partial \mathcal{L}}{\partial \varphi_i} = H(\mathbf{f}_i) - \lambda. \tag{11}$$

Since the optimal value of $\boldsymbol{\varphi}$ is constrained to either 1 or 0 for all samples, the closed-form solution to update φ_i is,

$$\varphi_i = \begin{cases} 1, & H(\mathbf{f}_i) \leq \lambda \\ 0, & \text{Otherwise} \end{cases}. \tag{12}$$

Algorithm 3 Joint training framework.

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, labels $\mathbf{Y}$, batch size B, iteration numbers T_1, T_2, self-paced hyper parameter λ.

Ensure:
      (1) Optimal node embeddings $\mathbf{E} \in \mathbb{R}^{n \times d}$;
      (2) Predicted probability distributions $\mathbf{F} \in \mathbb{R}^{n \times K}$.

1: Randomly initialize the parameters θ of GNN.
2: Initialize $\mathbf{F}$ by concatenating $\mathbf{Y}$ and zero matrix $\mathbf{0}^{(n-l) \times K}$.
3: Initialize $\boldsymbol{\varphi}$ as indicator vector with labeled nodes.
4: **while** not converge **do**
5:     **for** $t = 1, 2, \cdots, T_1$ **do**
6:         Sample B graph context pairs by Algorithm 1 with indicator vector $\boldsymbol{\varphi}$.
7:         Update the parameters by taking a gradient step based on (3).
8:     **end for**
9:     Obtain $\mathbf{E}$ and compute $\mathbf{S}$ based on (4) to fine-tune edge weights.
10:     **for** $t = 1, 2, \cdots, T_2$ **do**
11:         Update learned probability distribution $\mathbf{F}$ by taking a gradient step based on (10).
12:     **end for**
13:     Update $\boldsymbol{\varphi}$ based on (11) and (12).
14:     Augment λ.
15: **end while**

Note that, calculating the partial derivative of parameter set $\boldsymbol{\theta}$ in graph neural network module is quite easy, thus we omit the detailed mathematical derivations here due to space limitation. After we have obtained the derivatives of all the parameters, the whole optimization procedure can be efficiently performed via back-propagation. To sum up, the procedure of the joint training framework for CycProp model is depicted in Algorithm 3. First, the parameters $\mathbf{F}, \boldsymbol{\varphi}, \boldsymbol{\theta}$ are initialed. Then, in each iteration step, the graph neural network module and the label propagation module are updated for T_1 steps and T_2 steps respectively. In the end of an iteration step, the indicator $\boldsymbol{\varphi}$ is updated and the self-paced hyper parameter λ is increased. When the algorithm converges, the predicted classification results $\mathbf{F}$ and node embeddings $\mathbf{E}$ are returned simultaneously.

3.5 Complexity analysis

We analyze the time complexity of the proposed CycProp by considering the two main components respectively. For the graph neural network module, the time complexity can be compressed to $\mathcal{O}(|\mathcal{E}|) = \mathcal{O}(n\mathcal{D})$ with the sparse computation package where $|\mathcal{E}|$ is the number of edges and $\mathcal{D}$ is the average degree in a graph. For the label propagation module, the LPA has a linear complexity $\mathcal{O}(n)$, which is less than the graph neural network module

when the graphs are dense ($\mathcal{D}$ is large). Therefore, compared to other GNN-based methods, there is no significant increase in computation on CycProp.

4 Experiments

In this section, we report the results of our experiments to verify the effectiveness of our proposed CycProp model. We first describe the datasets and experimental setups in detail, and then we present the results with insights.

4.1 Datasets

We adopt three citation networks and two social networks for empirical studies. Statistics of the five datasets are summarized in Table 1 with more descriptions as follows,

- **Citation Networks**. Cora, Citeseer and Pubmed[1] [24] are three available public datasets, which are composed of scientific publications. In these networks, nodes represent published papers and edges denote citation relationships. Node labels indicate the categories to which each paper belongs and the text contents are treated as node attributes. We remove papers which have no connection in the network and extract the maximum connected component.
- **Social Networks**. Blogcatalog and Flickr[2] [34] are two typical social networks where nodes represent users and links denote the following relationships. In social networks, users usually generate personalized contents such as posting blogs or sharing photos with tag descriptions, thus these text contents are regarded as node attributes. We set the groups that users joined as labels, and users with no follower or predefined category have been removed.

4.2 Competitors

We compare the proposed CycProp model against several state-of-the-art baselines that can be categorized into the following groups:

- **Classical LPA.** These methods perform label propagation based on the edge weights calculated from the original attribute vectors. We consider three popular methods GFHF [49], LLGC [48] and DLP [37] as our compared algorithms.
- **Unsupervised Node Representation Learning.** Methods of this group first employ graph embedding techniques to learn the optimal node representations and then classify each node independently in the latent representation space. These approaches can be further classified into the following two classes:

    1) Structure-only. In this group, we choose three baselines DeepWalk [29], LINE [33] and node2vec [7], which utilize graph topological information only, and the node attributes are not taken into consideration.

    2) Attribute + Structure. This category of algorithms aims to encode both node attributes proximity and graph structure proximity into the latent representation space.

---

[1]http://linqs.cs.umd.edu/projects/projects/lbc

[2]http://socialcomputing.asu.edu/pages/datasets

Table 1 Statistics of the datasets. The information of the five benchmark datasets is given, including the number of nodes $\#|\mathcal{V}|$, the number of edges $\#|\mathcal{E}|$, the number of attributes $\#|Attrs|$, the number of labels $\#|L|$, and the average degree of graphs $\mathcal{D}$

Datasets	$\#|\mathcal{V}|$	$\#|\mathcal{E}|$	$\#|Attrs|$	$\#|L|$	Avg. Degree $\mathcal{D}$
Cora	2,485	5,069	1,433	7	2.04
Citeseer	2,110	3,719	3,703	6	1.76
Pubmed	19,717	44,338	500	3	2.25
Blogcatalog	5,196	171,743	8,189	6	33.05
Flickr	7,575	239,738	12,047	9	31.65

We consider three recently proposed methods SNE [20], EP [5] and GraphSAGE [8] as our baselines.

– **Semi-supervised Node Representation Learning.** Methods in this group further leverage additional label information to model the underlying representations. These approaches can be further classified into the following three classes:

1) Semi-supervised Node Embedding. This kind of methods train embeddings for each nodes with the supervision of label data. Planetoid [46] is one of the typical methods and is selected as the compared method.

2) GNN. By employing deep learning techniques [15], GNNs learn node representation as well as classifier in a joint and end-to-end way. We select two representative GNN models GCN [12] and GAT [35] as our baselines.

3) Neural Network with Label Propagation. This group of methods combine neural network with label propagation to enhance the classification performance. We choose three recently proposed methods NGM [3], GCN-LPA [39] and LP-DSSL [10] as our competitors.

For baseline algorithms, we use the source code released by the authors and the dimension of node embedding is set as 64 for all methods in all datasets. Specially, LP-DSSL [10] is originally designed with convolutional neural network for image classification. To adapt it to the network datasets, we replace the convolutional neural network with a two-layer GCN [12], and use the topology graph in dataset for label propagation instead of KNN graph. We randomly sample 30% labeled nodes as training set, and another 100 labeled nodes are sampled as a validation set to tune the hyper parameters. The remaining unlabeled nodes are used to test the performance of different algorithms. For our proposed CycProp model, the graph neural network's structure used in our experiments is a 2-hops neighborhood aggregation with dimension of 128 and 64, respectively. The sampled neighborhood size and negative context size are set as 10 for all datasets. We use rectified linear units as the activation function to introduce the non-linearity. To measure the classification result, we utilize both Micro-F1 (Mi-F1) and Macro-F1 (Ma-F1) as evaluation metrics. For unsupervised node representation methods, the learned node representations are regarded as features to train a one-vs-rest logistic regression classifier implemented by scikit-learn[3]. Evaluations are conducted by repeating 10 times with resampled labels, then the average score and its standard derivation are recorded as the final result.

---

[3]https://scikit-learn.org/stable/

4.3 Results and analysis

The experimental results of different algorithms over different datasets are presented in Table 2. To summarize, we have the following observations.

Generally, we can find that our proposed CycProp beats all baselines in all datasets for all settings. As expected, structure-only node representation learning methods like DeepWalk, LINE and node2vec perform worse than those approaches using node attributes (i.e, SNE, EP and GraphSAGE). The reason is that they only attempt to capture the graph's topology information, which provides very limited information compared to node attributes for node classification task. It's worth noting that these three classical LPAs such as LLGC, GFHF and DLP achieve better performance than structure-only node representation learning baselines. It further indicates the effectiveness of propagating labels on the graph, which gives us a solid foundation for the proposed CycProp model.

In addition, the semi-supervised methods consistently outperform unsupervised baselines with different gains by incorporating partially known node labels into the model. One major reason for the performance lift is because these semi-supervised methods are trained through an end-to-end manner, thus the learned node representations are specifically optimized for the classifier and show powerful discriminability. Finally, our proposed CycProp model is an efficient and direct way to learn the unknown node labels on graph, which aims to propagate labels rather than classifying each node independently. In classical two-step LPAs, edge weights are predetermined and cannot change during the learning process, thus its performance is bounded by the first step. To overcome these limitations, the proposed CycProp integrates GNN and label propagation in a unified framework via a mutually reinforcing manner that results in a great performance boost.

It is remarkable that compared with other methods that combine neural network and label propagation, CycProp shows better performance and generalization. Such a performance gap is caused by two reasons. First, while other methods that make predictions by neural networks, CycProp classifies each node by label propagation, which would not suffer from *over-smoothing* and *over-fitting* problems caused by GNN classifiers. Second, different from other methods that view label propagation as an auxiliary tool for GNNs (such as the regularization term in GCN-LPA or the pseudo-label generator in LP-DSSL), we treat these two algorithms as equal and mutually reinforcing components and integrate them into a joint learning framework. In this way, the advantages of both components are fully leveraged.

Besides, we can observe that CycProp has larger performance gains on the two social network datasets (BlogCatalog and Flickr). Specifically, compared to the baseline with the highest performance, the average Mi-F1 gain on social networks is 1.95%, while which on citation networks is 0.47%. The possible reason is that the social networks have a higher average degree (see Table 1). On these denser networks, the edge weights play a more important role in label propagation. Different from other methods, our proposed CycProp can adaptively optimize the edge weights, which bring significant performance gains on the dense networks.

4.4 Ablation study

In this subsection, we investigate how each of the two individual components contributes to the performance of CycProp. Figure 3 shows the classification results of CycProp and its two variants. Among them, CycProp-P denotes the case where only the label propagation module is used and CycProp-G indicates the case where only the graph neural network

Table 2 Classification performance on different datasets. Values in (·) are the standard derivation of multiple runs. The best performing method in each experiment is in bold

| Datasets	Cora		Citeseer		Pubmed		Blogcatalog		Flickr	
Metrics(%)	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
LLGC	77.6(0.7)	76.5(0.7)	62.9(0.8)	58.2(0.6)	82.6(0.3)	81.4(0.3)	65.8(0.4)	64.9(0.3)	54.3(0.3)	53.9(0.3)
GFHF	81.4(0.8)	80.7(0.5)	65.1(0.6)	61.4(0.5)	83.5(0.3)	82.7(0.2)	68.4(0.5)	67.7(0.4)	58.1(0.3)	57.7(0.2)
DLP	78.9(0.5)	77.6(0.4)	63.8(0.5)	59.7(0.4)	83.1(0.4)	82.4(0.3)	67.2(0.4)	66.6(0.4)	57.5(0.4)	57.1(0.3)
DeepWalk	75.8(0.5)	74.8(0.4)	57.3(0.6)	53.9(0.5)	81.6(0.2)	80.4(0.2)	64.5(0.5)	63.9(0.4)	48.7(0.4)	48.2(0.4)
LINE	73.6(0.6)	71.9(0.5)	57.1(0.7)	53.8(0.7)	81.2(0.4)	80.0(0.3)	64.9(0.4)	64.2(0.3)	51.4(0.4)	51.0(0.3)
node2vec	74.7(0.6)	72.5(0.4)	61.1(0.6)	56.5(0.4)	81.1(0.3)	79.6(0.3)	65.1(0.4)	64.5(0.4)	51.2(0.5)	49.8(0.4)
SNE	82.1(0.5)	81.5(0.5)	67.7(0.8)	63.3(0.6)	84.9(0.2)	83.5(0.2)	71.6(0.6)	70.8(0.5)	61.6(0.4)	61.1(0.4)
EP	82.5(1.2)	81.4(0.8)	68.5(1.4)	64.3(1.1)	83.8(0.7)	83.1(0.5)	73.5(1.3)	72.7(1.1)	63.3(0.9)	62.9(0.7)
GraphSAGE	83.6(0.9)	82.0(0.6)	69.2(0.8)	65.7(0.6)	84.5(0.4)	83.7(0.3)	75.8(0.6)	74.9(0.4)	66.2(0.5)	65.8(0.5)
Planetoid	84.2(1.0)	83.5(0.8)	73.6(0.5)	69.8(0.4)	85.5(0.3)	85.1(0.3)	78.4(0.3)	77.9(0.3)	68.1(0.4)	67.8(0.4)
GCN	85.5(1.1)	84.7(0.8)	77.5(0.8)	73.5(0.6)	85.6(0.2)	85.3(0.2)	79.2(0.5)	78.6(0.4)	69.5(0.2)	69.0(0.2)
GAT	86.1(0.8)	85.3(0.6)	77.9(0.7)	73.8(0.6)	86.4(0.3)	85.9(0.3)	80.6(0.4)	79.8(0.3)	71.4(0.3)	71.1(0.3)
NGM	83.9(0.7)	82.2(0.5)	72.9(0.6)	68.7(0.5)	85.1(0.3)	84.7(0.2)	77.1(0.4)	76.3(0.3)	67.7(0.6)	67.3(0.4)
GCN-LPA	86.5(0.6)	85.2(0.8)	77.7(0.8)	72.9(1.3)	86.1(0.8)	85.4(0.8)	79.6(0.7)	79.4(0.7)	69.2(0.6)	68.6(0.6)
LP-DSSL	87.4(0.6)	86.2(0.7)	78.1(1.1)	73.1(1.1)	87.5(0.7)	86.1(0.8)	79.8(0.6)	79.1(0.6)	69.9(0.4)	69.5(0.5)
CycProp	**87.6(0.9)**	**86.5(0.6)**	**79.1(0.8)**	**74.6(0.7)**	**87.7(0.2)**	**87.1(0.2)**	**82.3(0.3)**	**81.6(0.3)**	**73.6(0.3)**	**73.2(0.3)**
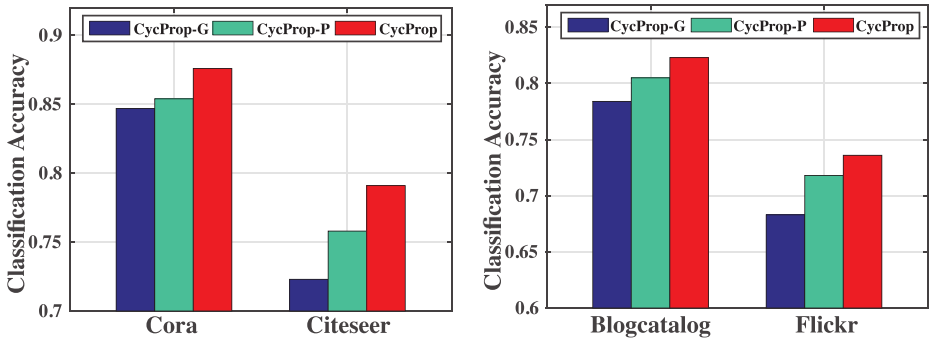
Figure 3 The performance of CycProp and its variants. CycProp-P and CycProp-G indicate the variants only using label propagation module and graph neural network module respectively

module is used. The best results are achieved by the full CycProp, which validates the effectiveness of combining these two modules in a mutually reinforcing manner. Moreover, we can observe that CycProp-P always outperforms CycProp-G, which indicates the propagation module seems to play a more important role in the joint framework. A possible reason is that the label propagation can efficiently prevent the shortages of GNNs, e.g., over-fitting and over-smoothing, especially when the training labels are scarce. This result also verifies the advantage of propagating labels on graph rather than classifying each node independently.

4.5 Parameter sensitivity

In this subsection, we study the impact of several parameters by varying them in different scales. Due to the limited space, we only show sensitivities of the trade-off parameter α of (7) and the node embedding dimension d in Figure 4. As we can see, $\alpha = 0.1$ is mostly the best across different data sets. When α is too small or too large, the performance becomes worse. For node embedding dimension d, we observe that, by increasing d, the performance first increases and then keeps stable. Besides, λ is initialized with 0.1 and $\mu = 10$, $\delta = 0.1$ usually get the best results and we don't observe too much difference when varying them.
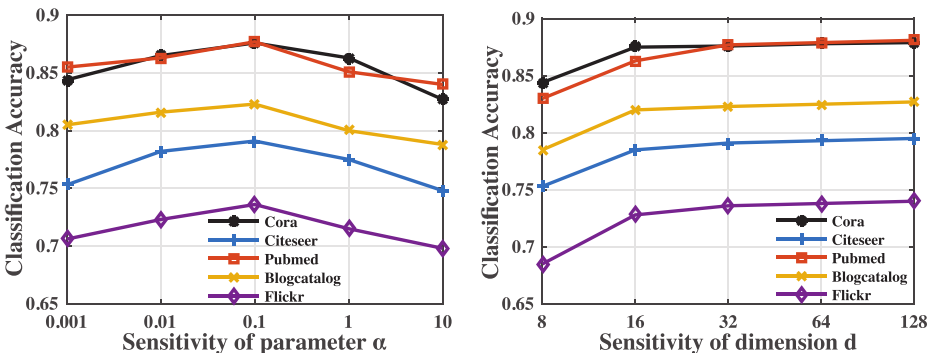

Figure 4 The sensitivity of the trade-off parameter α and the node embedding dimension d

5 Conclusions

In this paper, we investigate the semi-supervised learning task on graphs and introduce a unified framework CycProp, which integrates label propagation and GNN in a cyclic and dynamically reinforcing manner. Specifically, in each iteration, we employ the graph neural network module to learn the informative node embeddings that can refine the edge weights for facilitating the label propagation; then the learned highly reliable labels obtained from the label propagation module are incorporated into the model to fine-tune the node embedding procedure, thus forming a closed cyclic training loop. Extensive experiments on five real-world datasets demonstrate the effectiveness of CycProp and its superiority to a range of state-of-the-art methods. The most significant advantage of the proposed CycProp framework is that it avoids the shortages of both GNNs and LPA and leverages the goodness of both of them. However, CycProp relies on an alternate updating to learn both of the components, which is practical but might result in a suboptimal solution. We will overcome this disadvantage in our future study. The future works are mainly two-fold. First, we plan to extend the idea of combining GNNs and LPA into heterogeneous graphs where nodes and links are of different types. Second, we will investigate the deeper integration of GNNs and LPA based on the advanced techniques in each type of method, which focuses on solving the aforementioned suboptimal problem.

References

1. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. JMLR **7**(Nov), 2399–2434 (2006)
2. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: COMPSTAT, pp. 177–186. Springer (2010)
3. Bui, T.D., Ravi, S., Ramavajjala, V.: Neural graph learning: Training neural networks using graphs. In: WSDM, pp. 64–71. ACM (2018)
4. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: AAAI, pp. 3438–3445 (2020)
5. Duran, A.G., Niepert, M.: Learning graph representations with embedding propagation. In: NIPS, pp. 5119–5130 (2017)
6. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: ICML (2017)
7. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD, pp. 855–864. ACM (2016)
8. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS, pp. 1024–1034 (2017)
9. Huang, X., Li, J., Hu, X.: Accelerated attributed network embedding. In: SDM, pp. 633–641. SIAM (2017)
10. Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Label propagation for deep semi-supervised learning. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pp. 5070–5079. Computer Vision Foundation / IEEE (2019). https://doi.org/10.1109/CVPR.2019.00521
11. Jacob, Y., Denoyer, L., Gallinari, P.: Learning latent representations of nodes for classifying in heterogeneous social networks. In: WSDM, pp. 373–382. ACM (2014)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
13. Kudo, T., Maeda, E., Matsumoto, Y.: An application of boosting to graph classification. In: NIPS, pp. 729–736 (2005)
14. Kumar, M.P., Packer, B., Koller, D.: Self-paced learning for latent variable models. In: NIPS, pp. 1189–1197 (2010)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)

16. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical comparison of algorithms for network community detection. In: WWW, pp. 631–640. ACM (2010)
17. Li, H., Lin, Z.: Accelerated proximal gradient methods for nonconvex programming. In: NIPS, pp. 379–387. MIT Press (2015)
18. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
19. Liang, J., Jacobs, P., Sun, J., Parthasarathy, S.: Semi-supervised embedding in attributed networks with outliers. In: Proceedings of the 2018 SIAM International Conference on Data Mining, pp. 153–161. SIAM (2018)
20. Liao, L., He, X., Zhang, H., Chua, T.S.: Attributed social network embedding. In: arXiv:1705.04969 (2017)
21. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American Society for Information Science and Technology **58**(7), 1019–1031 (2007)
22. Liu, L., Zhou, T., Long, G., Jiang, J., Dong, X., Zhang, C.: Isometric propagation network for generalized zero-shot learning. In: International Conference on Learning Representations (ICLR) (2021)
23. Liu, L., Zhou, T., Long, G., Jiang, J., Zhang, C.: Attribute propagation network for graph zero-shot learning. In: AAAI Conference on Artificial Intelligence (AAAI) (2020)
24. Lu, Q., Getoor, L.: Link-based classification. In: ICML, pp. 496–503 (2003)
25. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR (2013)
26. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
27. Pan, S., Wu, J., Zhu, X., Zhang, C., Wang, Y.: Tri-party deep network representation. In: IJCAI, pp. 1895–1901 (2016)
28. Parikh, N., Boyd, S.: Proximal algorithms. Foundations and Trends in Optimization **1**(3), 127–239 (2014)
29. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: SIGKDD, pp. 701–710. ACM (2014)
30. Rong, Y., Huang, W., Xu, T., Huang, J.: Dropedge: Towards deep graph convolutional networks on node classification. In: International Conference on Learning Representations (2019)
31. Shi, W., Rajkumar, R.: Point-gnn: Graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1711–1719 (2020)
32. Tang, J., Qu, M., Mei, Q.: Pte: Predictive text embedding through large-scale heterogeneous text networks. In: SIGKDD, pp. 1165–1174. ACM (2015)
33. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW, pp. 1067–1077. ACM (2015)
34. Tang, L., Liu, H.: Relational learning via latent social dimensions. In: SIGKDD, pp. 817–826. ACM (2009)
35. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
36. Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. In: International Conference on Learning Representations (2019)
37. Wang, B., Tu, Z., Tsotsos, J.K.: Dynamic label propagation for semi-supervised multi-class multi-label classification. In: ICCV, pp. 425–432 (2013)
38. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. In: ICML, pp. 985–992. ACM (2006)
39. Wang, H., Leskovec, J.: Unifying graph convolutional neural networks and label propagation. arXiv:2002.06755 (2020)
40. Wang, W., Carreira-Perpinán, M.A.: Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. In: arXiv:1309.1541 (2013)
41. Wu, F., Souza, J.r., A. H., Zhang, T., Fifty, C., Yu, T., Weinberger, K.Q.: Simplifying graph convolutional networks. In: ICML (2019)
42. Wu, L., Wang, D., Feng, S., Zhang, Y., Yu, G.: Mdal: Multi-task dual attention lstm model for semi-supervised network embedding. In: International Conference on Database Systems for Advanced Applications, pp. 468–483. Springer (2019)
43. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems (2020)
44. Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C.: Connecting the dots: Multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 753–763 (2020)

45. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2019)
46. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: ICML, pp. 40–48. JMLR.org (2016)
47. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: GNNExplainer: Generating explanations for graph neural networks. In: Advances in neural information processing systems, pp. 9244–9255 (2019)
48. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: NIPS, pp. 321–328 (2004)
49. Zhu, X., Ghahramani, Z., Lafferty, J.D.: Semi-supervised learning using Gaussian fields and harmonic functions. In: ICML, pp. 912–919 (2003)