Network Embedding with Completely-imbalanced Labels

Zheng Wang, Xiaojun Ye, Chaokun Wang, Jian Cui, and Philip S. Yu, *Fellow, IEEE*

Abstract— Network embedding, aiming to project a network into a low-dimensional space, is increasingly becoming a focus of network research. Semi-supervised network embedding takes advantage of labeled data, and has shown promising performance. However, existing semi-supervised methods would get unappealing results in the **completely-imbalanced** label setting where some classes have no labeled nodes at all. To alleviate this, we propose two novel semi-supervised network embedding methods. The first one is a shallow method named RSDNE. Specifically, to benefit from the completely-imbalanced labels, RSDNE guarantees both intra-class similarity and inter-class dissimilarity in an approximate way. The other method is RECT which is a new class of graph neural networks. Different from RSDNE, to benefit from the completely-imbalanced labels, RECT explores the knowledge of class-semantic descriptions. This enables RECT to handle networks with node features and multi-label setting. Experimental results on several real-world datasets demonstrate the superiority of the proposed methods.

Index Terms—Network embedding, Graph neural networks, Social network analysis, Data mining.

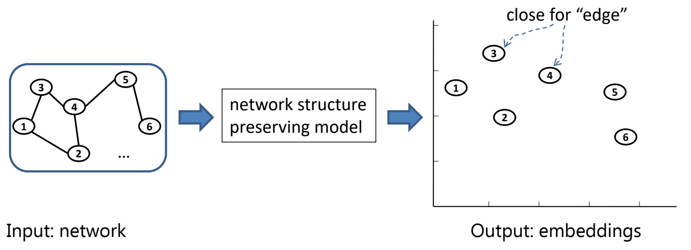
1 Introduction

NETWORK analysis [1] [2] [3] [4] is a hot research topic in various scientific areas like social science, computer science, biology and physics. Many algorithmic tools for network analysis heavily rely on network representation which is traditionally represented by the adjacency matrix. However, this straightforward representation not only lacks of representative power but also suffers from the data sparsity issue [5].

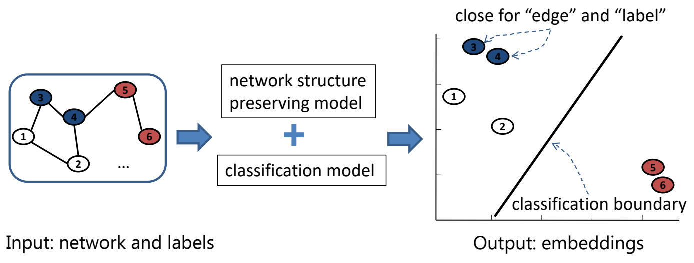
Recently, learning dense and low-dimensional vectors as representations for networks has aroused considerable research interest in network analysis. It has been shown that the learned representations could benefit many network analysis tasks, such as node classification [6], link prediction [7] [8] and network visualization [9]. Commonly, learning network representation is also known as network embedding [10]. The learned low-dimensional vectors are called node embeddings (or representations).

One basic requirement of network embedding is to preserve the inherent network structure in the embedding space, as illustrated in Fig. 1(a). Early studies, like IsoMap [11] and LLE [12], ensure the embedding similarity among linked nodes. Now, more research activities focus on preserving the unobserved but legitimate links in the network. For example, DeepWalk [6] exploits the node co-occurring relationships in the truncated random walks over a network. LINE [13] considers both the first-order and second-order proximities of a network. Unlike the above two shallow methods, SDNE [14], a class of graph neural networks (GNNs) [15] [16], uses multiple layers of non-linear functions to model these two proximities.

Semi-supervised network embedding methods, which take advantage of labeled data, have shown promising performance. Typical semi-supervised shallow methods include LSHM [17], LDE [18], and MMDW [19]. Typical semi-supervised GNNs are GCN [20], GAT [21] and APPNP [22]. As illustrated in Fig. 1(b), in these methods, a classification model (e.g., SVM [23] and Cross-entropy [24]) will be learned to inject label information. Intuitively, in the embedding space, the learned classification model would reduce



(a) Unsupervised network embedding

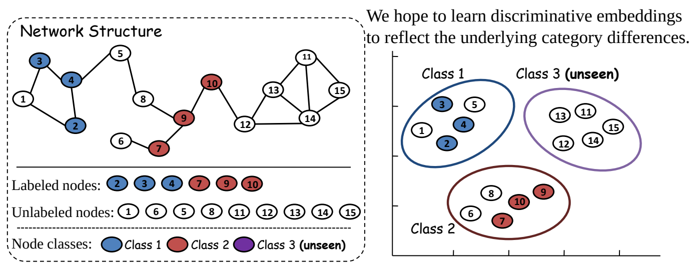

close for "edge" and "label"
network structure preserving model
+
classification model
classification boundary
Input: network and labels
Output: embeddings

(b) Semi-supervised network embedding

Fig. 1: Frameworks of existing unsupervised and semi-supervised network embedding methods.

• Z. Wang and J. Cui are with the Department of Computer Science and Technology, University of Science and Technology Beijing, China.
E-mail: wangzheng@ustb.edu.cn, g20178653@xs.ustb.edu.cn.
• X. Ye and C. Wang are with the School of Software, Tsinghua University, Beijing, China.
E-mail: {yexj, chaokun}@mail.tsinghua.edu.cn.
• P.S. Yu is with the Department of Computer Science, University of Illinois at Chicago IL 60607. E-mail: psyu@cs.uic.edu.

Manuscript received April xx, 20xx; revised xx, 20xx.


Network Structure
Labeled nodes:
Unlabeled nodes:
Node classes: Class 1 Class 2 Class 3 (unseen)
We hope to learn discriminative embeddings to reflect the underlying category differences.
Class 1
Class 3 (unseen)
Class 2

(a) Input: network and labels (b) Output: embeddings

Fig. 2: Illustration of the semi-supervised network embedding with completely-imbalanced labels. This toy network actually contains three classes of nodes, but only two classes provide labeled nodes, i.e., blue and red nodes. The remaining nodes (including all the nodes of Class 3) are unlabeled.

the distance between same labeled nodes and enlarge the distance between different labeled nodes. Influenced by this, the embedding results therefore become more discriminative and have shown state-of-the-art performance.

1.1 Problem

Most semi-supervised network embedding methods [17] [18] [19] assume the labeled data is generally balanced, i.e., every class has at least one labeled node. In this paper, we consider a more challenging scenario in which some classes have no labeled nodes at all (shown in Fig. 2), i.e., the **completely-imbalanced** case. This problem can be formulated as follows:

Problem (Network embedding with completely-imbalanced labels). *Given a network $\mathcal{G} = (\mathcal{V}, A, \mathcal{C}, \mathcal{C}^s)$ where $\mathcal{V}$ is the set of n nodes, $A \in \mathbb{R}^{n\times n}$ is the adjacency matrix, $\mathcal{C}$ is the whole node class label set, and $\mathcal{C}^s \subset \mathcal{C}$ is the observed label set, our goal is to learn a continuous low-dimensional vector $u_i \in \mathbb{R}^d$ ($d \ll n$) for each node v_i, such that nodes close to each other in the network structure and with the similar class labels are close in the embedding space.*

This problem deserves special attention for two reasons. Firstly, it has many practical applications. For example, considering Wikipedia which can be seen as a set of linked web pages on various topics [25], it is difficult to collect labeled samples for every topic exactly and not miss any one. Secondly, and more importantly, without considering this issue, traditional semi-supervised methods would yield unappealing results. To verify this, we carry out an experiment on Citeseer dataset [26], in which the nodes from unseen classes are excluded from the labeled data. We test two typical semi-supervised methods (i.e., a shallow method LSHM and a GNN method GCN) on node classification task. As shown in Table 1, their performance declines noticeably compared with their counterparts trained with the balanced labels. This decline might be caused by the classification models used in these methods, since general classifiers are very likely to get biased results on imbalanced data [27]. We refer to Sections 5 and 6 for more detailed discussion.

TABLE 1. Classification performance on Citeseer. Here: we use $\mathcal{M}(b)$ and $\mathcal{M}(\text{-}t)$ to denote the method $\mathcal{M}$ using the balanced and completely-imbalanced labeled data with t unseen classes, respectively.

		Accuracy			Relative Accuracy Decline		
Method	*Label*	10%	30%	50%	10%	30%	50%
LSHM	LSHM(b)	0.5007	0.6178	0.6711	-	-	-
	LSHM(-1)	0.4258	0.5887	0.6455	0.1496↓	0.0471↓	0.0382↓
	LSHM(-2)	0.4253	0.5504	0.6027	0.1506↓	0.1091↓	0.1019↓
GCN	GCN(b)	0.7198	0.7473	0.7628	-	-	-
	GCN(-1)	0.6572	0.6937	0.7064	0.0870↓	0.0717↓	0.0739↓
	GCN(-2)	0.4761	0.5085	0.5159	0.3386↓	0.3196↓	0.3237↓

1.2 Contribution

To address this problem, in this paper, we first present a novel shallow method termed RSDNE. The basic idea is to guarantee both intra-class similarity and inter-class dissimilarity in an approximate way, so as to benefit from completely-imbalanced labels. Specifically, we relax the intra-class similarity requirement by allowing the same labeled nodes to lie on the same manifold in the embedding space. On the other hand, we approximate the inter-class dissimilarity requirement by removing the known connections between the nodes with different labels. As such, our method can reasonably guarantee these two requirements and also avoid the biased results. We further formalize these approximations into a unified embedding framework, and give an efficient learning algorithm.

To leverage the power of deep neural networks [28], we further propose RECT, a new class of GNNs. Comparing to RSDNE, RECT can leverage node features and deal with the multi-label case [29]. In particular, to utilize the completely-imbalanced labels, unlike RSDNE nor traditional GNNs, RECT adopts a novel objective function which explores the class-semantic knowledge. This is motivated by the recent success of Zero Shot Learning (ZSL) [30], which has demonstrated the ability of recognizing unseen objects via introducing class-semantic descriptions. In addition, unlike the traditional ZSL methods, the class-semantic descriptions used in RECT do not rely on human annotations or any third-party resources, making RECT well suited for practical applications.

In summary, our main contributions are as follows:

1) We study the problem of network embedding with completely-imbalanced labels. To our best knowledge, little work has addressed this problem.
2) We propose an effective shallow method named RSDNE which can learn discriminative embeddings by approximately guaranteeing both intra-class similarity and inter-class dissimilarity.
3) We propose RECT a new class of graph neural networks. Comparing to RSDNE, RECT can further handle networks with node features and multi-label setting.
4) We conduct extensive experiments on five real-world datasets in both completely-imbalanced setting and balanced setting to demonstrate the superiority of our methods.

In addition, it is worth highlighting that in the balanced label setting, our methods could still achieve comparable performance to state-of-the-art semi-supervised methods, although our methods are not specially designed for this setting. Therefore, our methods would be favorably demanded by the scenario where the quality of labels cannot be guaranteed.

The remainder of this paper is organized as follows. We review some related work in Section 2. In Section 3, we elaborate our shallow method RSDNE with details. In Section 4, we introduce the proposed GNN method RECT. Section 5 discusses the rationality of our methods, and further analyzes the relationship between the existing methods and ours. Section 6 reports experimental results. Section 7 concludes this paper.

2 Related Work

2.1 Semi-supervised Network Embedding

The goal of semi-supervised network embedding is to learn the representations of both labeled and unlabeled nodes. Existing shallow methods mainly share the similar idea, that is, to jointly train a network structure preserving model and a class classification model. For example, LDE [18] considers the first-order proximity [13] of the network and jointly trains a 1-nearest neighbor classification model [31]. Semi-supervised GNNs also will train a classification model but explicitly preserve the network structure information. In particular, most GNNs (like GCN [20], GAT [21] and APPNP [22]) iteratively perform feature aggregations based on the network structure [32]. We refer readers to a comprehensive survey [33] for more discussions.

However, these methods all assume the labeled data is generally balanced (i.e., label information covers all classes), otherwise would get unappealing results. In practice, the quality of labeled data is hard to guarantee. Therefore, to enhance the applicability, we investigate network embedding in the completely-imbalanced label setting.

2.2 Imbalanced Data Learning

A training dataset is called imbalanced if at least one of the classes are represented by significantly less number of instances than the others. The imbalanced data are pervasively existed in multiple domains ranging from the physical world to social networks, and to make proper use of such data is always a pivotal challenge [34] [35]. This topic has been identified in several vital research areas, such as classification [36], clustering [37], and data streams [38]. We refer to [27] and [39] for a comprehensive survey. However, in the area of network embedding, little previous work considers the imbalanced problem, not to mention the completely-imbalanced problem [10].

2.3 Zero Shot Learning

ZSL [40] [41], which is recently a hot research topic in computer vision, aims to recognize the objects from unseen classes. To achieve this goal, it leverages some high-level semantic descriptions (also called as attributes) shared between both seen and unseen classes. For example, we can define some attributes like “wing”, “climb” or “tail” for animals. Then we can train attribute recognizers using images and attribute information from seen classes. After that, given an image from unseen classes, we can infer its attributes. By comparing the difference between the inferred attributes and each unseen classes’ attributes, the final output is given based on the score. Generally, attributes are human annotated, which needs lots of human efforts. Another more practical way is to use word embeddings generated by word2vec tools [42] trained with large-scale general text database. Despite of this, attributes collection still heavily relies on third-party resources, limiting the use of ZSL methods in practical applications. Additionally, although various ZSL methods have been proposed [43], all these methods are limited to classification or prediction scenario. To our best knowledge, there is few reported work considering the unseen classes in the network embedding problem.

3 The Proposed Shallow Method: RSDNE

In this section, we first introduce a network structure preserving model. Then, we present our method with another two objective terms for completely-imbalanced labels. Finally, we give an efficient optimization algorithm.

3.1 Modeling Network Structure with DeepWalk

To capture the topological structure of a network, DeepWalk performs random walks over a network to get node sequences. By regarding each node sequence $\omega = \{v_1, ..., v_{|\omega|}\}$ as a word sequence, it adopts the well-known language model Skip-Gram [42] to maximize the likelihood of the surrounding nodes given the current node v_i for all random walks $\omega \in \Omega$:

$$\sum_{\omega\in\Omega}[\frac{1}{|\omega|}\sum_{i=1}^{|\omega|}\sum_{-r\leq j\leq r}\log Pr(v_{i+j}|v_i)] \tag{1}$$

where r is the radius of the surrounding window, and the probability $Pr(v_j|v_i)$ is obtained via the softmax:

$$Pr(v_j|v_i) = \frac{exp(u_j \cdot u_i)}{\sum_{t\in\mathcal{V}} exp(u_t \cdot u_i)} \tag{2}$$

where u_i is the representation vector of node v_i, and $\cdot$ is the inner product between vectors.

Yang et al. [44] has proved that DeepWalk actually factorizes a matrix M whose entry M_{ij} is formalized as:

$$M_{ij} = \log\,[e_i(\bar{A} + \bar{A}^2 + \cdots + \bar{A}^t)]/t \tag{3}$$

where $\bar{A}$ is the transition matrix which can be seen as a row normalized network adjacency matrix, and e_i denotes an indicator vector whose i-th entry is 1 and the others are all 0. To balance speed and accuracy, [44] finally factorized the matrix $M{=}(\bar{A}{+}\bar{A}^2)/2$ instead, since sparse matrix multiplication can be easily parallelized and efficiently calculated [45].

More formally, the matrix factorization model of DeepWalk aims to find a (node embedding) matrix $U \in \mathbb{R}^{n\times d}$ and a (context embedding) matrix $H \in \mathbb{R}^{d\times n}$ via solving the following optimization problem:

$$\min_{U,H}\ \mathcal{J}_{DW}{=}\,\|M - UH\|_F^2 + \lambda(\|U\|_F^2 + \|H\|_F^2) \tag{4}$$

where λ is the regularization parameter to avoid overfitting. In this paper, we adopt this model (i.e., Eq. 4) as our basic network structure preserving model.

3.2 Modeling Intra-class Similarity

In this completely-imbalanced setting, the labeled nodes all come from the seen classes. Intuitively, we should ensure the *intra-class similarity*, i.e., the nodes sharing the same label should be close to each other in the embedding space. To satisfy this, traditional semi-supervised methods employ various classifiers to reduce the intra-class embedding variance. However, this would yield unappealing results with completely-imbalanced labels (shown in Table 1).

To alleviate this, we relax this similarity requirement by allowing the same labeled nodes to lie on the same manifold, i.e., a topological space which can be Euclidean only locally [12]. Although the underlying manifold is unknown, we can build a sparse adjacency graph to approximate it [46]. In other words, each labeled node only needs to be close to k ($k \ll n$, and k=5 in our experiments) same labeled nodes. However, we do not know how to select the best k nodes, since the optimal node alignments in the new embedding space is unknown. A simple solution is to randomly select k same labeled nodes, which may not be optimal.

In this paper, we solve this problem in an adaptive way. For notational convenience, for a labeled node v_i, we call the selected k nodes as v_i's *intra-class neighbors*. Suppose we use $S \in \{0,1\}^{n\times n}$ to denote the intra-class neighbor relationship among nodes, i.e., S_{ij}=1 when node v_j is the intra-class neighbor of node v_i, otherwise S_{ij}=0. Mathematically, S can be obtained by solving the following optimization problem:

$$\begin{aligned}\min_{U,S}\ & \mathcal{J}_{intra} = \frac{1}{2}\sum_{i,j=1}^{n} \|u_i - u_j\|_F^2 S_{ij} \\ \text{s.t. } & \forall i \in \mathcal{L}, s_i'\mathbf{1} = k,\ S_{ii} = 0 \\ & \forall i,j \in \mathcal{L}, S_{ij} \in \{0,1\},\ \text{if } \mathcal{C}_i^s = \mathcal{C}_j^s \\ & \forall i,j, S_{ij} = 0,\ \text{if } i \notin \mathcal{L} \text{ or } \mathcal{C}_i^s \neq \mathcal{C}_j^s\end{aligned} \tag{5}$$

where $\mathcal{L}$ is the labeled node set, and $s_i \in R^{n\times 1}$ is a vector with the j-th element as S_{ij} (i.e., s_i', the transpose of s_i, is the row vector of matrix S), and $\mathbf{1}$ denotes a column vector with all entries equal to one, and $\mathcal{C}_i^s$ and $\mathcal{C}_j^s$ are the (seen) class labels of node v_i and v_j respectively. In this paper, $(\cdot)'$ stands for the transpose.

3.3 Modeling Inter-class Dissimilarity

Although Eq. 5 models the similarity within the same class, it neglects the *inter-class dissimilarity*, i.e., the nodes with different labels should be far away from each other in the embedding space. Traditional semi-supervised methods employ different classification models to enlarge the inter-class embedding variance. Nevertheless, this would yield unappealing results with completely-imbalanced labels (shown in Table 1).

To alleviate this, we approximate this dissimilarity requirement by removing the known connections between the nodes with different labels. Since we adopt the matrix form of DeepWalk (i.e., matrix M in Eq. 4) to model the connections among nodes, this approximation leads to the following optimization problem:

$$\min_{U}\ \mathcal{J}_{inter} = \frac{1}{2}\sum_{i,j=1}^{n} \|u_i - u_j\|_F^2 W_{ij} \tag{6}$$

where W is a weighted matrix whose element W_{ij}=0 when labeled nodes v_i and v_j belong to different categories, otherwise $W_{ij} = M_{ij}$.

3.4 The Unified Model: RSDNE

With modeling the network structure (Eq. 4), intra-class similarity (Eq. 5) and inter-class dissimilarity (Eq. 6), the proposed method is to solve the following optimization problem:

$$\begin{aligned}\min_{U,H,S}\ & \mathcal{J} = \mathcal{J}_{DW} + \alpha(\mathcal{J}_{intra} + \mathcal{J}_{inter}) \\ \text{s.t. } & \forall i \in \mathcal{L}, s_i'\mathbf{1} = k,\ S_{ii} = 0 \\ & \forall i,j \in \mathcal{L}, S_{ij} \in \{0,1\},\ \text{if } \mathcal{C}_i^s = \mathcal{C}_j^s \\ & \forall i,j, S_{ij} = 0,\ \text{if } i \notin \mathcal{L} \text{ or } \mathcal{C}_i^s \neq \mathcal{C}_j^s\end{aligned} \tag{7}$$

where α is a balancing parameter. Since both the relaxed similarity and dissimilarity requirements of labels have been considered, we call the proposed method as **R**elaxed **S**imilarity and **D**issimilarity **N**etwork **E**mbedding (RSDNE).

A Light Version of RSDNE: For each labeled node v_i, to identify its optimal k intra-class neighbors, RSDNE needs to consider all the nodes which have the same label with v_i. This would become inefficient when more labeled data is available (some theoretical analysis can be found in Section 5.2). Therefore, we give a light version of RSDNE (denoted as RSDNE*). The idea is that: for a labeled node v_i, at the beginning, we can randomly select $\bar{k}$ ($k < \bar{k} \ll n$) same labeled nodes to gather v_i's intra-class neighbor candidate set $\mathcal{O}_i$. Based on this idea, this light version RSDNE* is to solve the following optimization problem:

$$\begin{aligned}\min_{U,H,S}\ & \mathcal{J} = \mathcal{J}_{DW} + \alpha(\mathcal{J}_{intra} + \mathcal{J}_{inter}) \\ \text{s.t. } & \forall i \in \mathcal{L}, s_i'\mathbf{1} = k,\ S_{ii} = 0 \\ & \forall i \in \mathcal{L}, j \in \mathcal{O}_i, S_{ij} \in \{0,1\} \\ & \forall i,j, S_{ij} = 0,\ \text{if } i \notin \mathcal{L} \text{ or } \mathcal{C}_i^s \neq \mathcal{C}_j^s\end{aligned} \tag{8}$$

3.5 Optimization

3.5.1 Optimization for RSDNE

The objective function in Eq. 7 is a standard quadratic programming problem with 0/1 constraints, which might be difficult to solve by the conventional optimization tools. In this study, we propose an efficient alternative optimization strategy for this problem.

Update U As Given H and S: When S is fixed, the objective function in Eq. 5 can be rewritten as $Tr(U'L_sU)$, where $L_s = D_s - (S+S')/2$ and D_s is a diagonal matrix whose i-th diagonal element is $\sum_j (S_{ij}+S_{ji})/2$. Similarly, the objective function in Eq. 6 can be rewritten as $Tr(U'L_wU)$ where $L_w = D_w - (W+W')/2$ and D_w is a diagonal matrix whose i-th diagonal element is $\sum_j (W_{ij}+W_{ji})/2$. As such, when H and S are fixed, problem (7) becomes:

$$\min_{U} \mathcal{J}_U = \|M - UH\|_F^2 + \alpha(Tr(U'L_sU) + Tr(U'L_wU)) + \lambda\|U\|_F^2 \tag{9}$$

The derivative of $\mathcal{J}_U$ w.r.t. U is:

$$\frac{\partial \mathcal{J}_U}{\partial U} = 2(-MH' + UHH' + \alpha(L_s + L_w)U + \lambda U) \tag{10}$$

Algorithm 1 RSDNE

```
Require: Matrix form of DeepWalk $M$, label information,
    learning rate $\eta$, and parameters $\alpha$ and $\lambda$ ;
Ensure: The learned network node embedding result $U$;
1: Initialize $U$, $H$ and $S$;
2: repeat
3:     Update $U$ by $U = U - \eta\frac{\mathcal{J}_U}{\partial U}$;
4:     Update $H$ by $H = H - \eta\frac{\mathcal{J}_H}{\partial H}$;
5:     Update $S$ by solving problem (13) ;
6:     Change the learning rate $\eta$ according to some rules,
    such as Armijo [47];
7: until Convergence or a certain iterations;
8: return $U$.
```

Update H As Given U and S: When U and S are fixed, problem (7) becomes:

$$\min_{H} \ \mathcal{J}_H = \|M - UH\|_F^2 + \lambda \|H\|_F^2 \tag{11}$$

The derivative of $\mathcal{J}_H$ w.r.t. H is:

$$\frac{\partial \mathcal{J}_H}{\partial H} = 2(-U'M + U'UH + \lambda H) \tag{12}$$

Update S As Given U and H: When U and H are fixed, problem (7) becomes:

$$\begin{aligned} \min_{S} \ \mathcal{J}_S &= \frac{\alpha}{2}\sum_{i,j=1}^{n} \|u_i - u_j\|_F^2 S_{ij} \\ \text{s.t. } &\forall i \in \mathcal{L}, s_i'\mathbf{1} = k, \ S_{ii} = 0 \\ &\forall i,j \in \mathcal{L}, S_{ij} \in \{0,1\}, \text{ if } \mathcal{C}_i^s = \mathcal{C}_j^s \\ &\forall i,j, S_{ij} = 0, \text{ if } i \notin \mathcal{L} \text{ or } \mathcal{C}_i^s \neq \mathcal{C}_j^s \end{aligned} \tag{13}$$

As problem (13) is independent between different i, we can deal with the following problem individually for each labeled node v_i [1]:

$$\begin{aligned} \min_{s_i, i\in\mathcal{L}} \ &\sum_{j=1}^{n} \|u_i - u_j\|_F^2 S_{ij} \\ \text{s.t. } &s_i'\mathbf{1} = k, \ S_{ii} = 0 \\ &\forall j, S_{ij} = 0, \text{ if } j \notin \mathcal{L} \\ &\forall j \in \mathcal{L}, S_{ij} \in \{0,1\}, \text{ if } \mathcal{C}_i^s = \mathcal{C}_j^s \end{aligned} \tag{14}$$

The optimal solution to problem (14) is (proved in Section 5.1):

$$S_{ij} = \begin{cases} 1, & \text{if } v_j \in \mathcal{N}_{kc}(v_i); \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

where set $\mathcal{N}_{kc}(v_i)$ contains the top-k nearest and same labeled nodes to v_i in the current calculated embedding space.

For clarity, we summarize the complete RSDNE algorithm for network embedding in Alg. 1.

3.5.2 Optimization for RSDNE*

The optimization approach for RSDNE* is almost the same as Alg. 1. The only difference is that: when updating S as given U and H, for each labeled node v_i, we only need to sort the nodes in (it's intra-class neighbor candidate set) $\mathcal{O}_i$ to get the top-k nearest and same labeled neighbors, so as to get the optimal solution of S.

1. For an unlabeled node v_i, the solution is $s_i' = 0$.

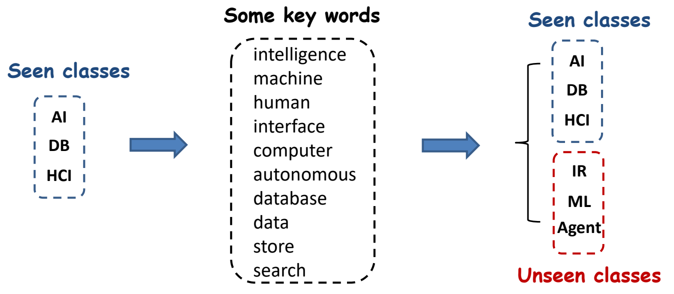


Fig. 3: Some words sampled from the documents of three seen classes (i.e., AI, DB, and HCI) in Citeseer.

4 THE PROPOSED GNN METHOD: RECT

It is inappropriate to directly adopt the objective function of RSDNE (Eq. 7 or Eq. 8) into traditional neural networks (like multilayer perceptron) which are not suited for graph-structured[2] data. Moreover, simultaneously optimizing multiple objective terms is a challenging engineering task, and usually results in a degenerate solution [48]. In this section, we first give a brief introduction to GNN, and then propose a novel effective and easy-to-implement GNN method.

4.1 Preliminaries: Graph Neural Network

GNN [16] is a type of neural network model for graph-structured data. Generally, GNN models are dynamic models where the hidden representation of all nodes evolve over layers. Given a graph with the adjacent matrix A, at the t-th hidden layer, the representation $z_{v_i}^t$ for node v_i is commonly updated as follows:

$$\begin{aligned} b_{v_i}^t &= \mathcal{F}_b(\{z_{v_j}^t | v_j \in \Psi_{v_i}\}) \\ z_{v_i}^{t+1} &= \mathcal{F}_z(\{b_{v_i}^t, z_{v_i}^t\}) \end{aligned} \tag{16}$$

where $b_{v_i}^t$ is a vector indicating the aggregation of messages that node v_i receives from its neighbors Ψ_{v_i}. Function $\mathcal{F}_b$ is a message calculating function, and $\mathcal{F}_z$ is a hidden state update function. Similar to the common neural networks, $\mathcal{F}_b$ and $\mathcal{F}_z$ are feed-forward neural layers. By specifying these two functional layers, we can get various GNN variants, like Graph convolutional network (GCN) [20] and Graph attention network (GAT) [21].

To inject label information, GNNs usually end up with a softmax layer to train a classification model. Once the training of GNNs is completed, the outputs of any intermedian layers can be adopt as the final graph embedding results. However, as shown in Table 1, this kind of methods will yield unappealing results in completely-imbalanced label setting. The fundamental cause is that the known supervised information only reflects the knowledge of seen classes but ignores that of unseen classes. Therefore, in the completely-imbalanced setting, the key issue is: how to deduce the supervised information, which contains both the knowledge of seen and unseen classes, from the limited labeled nodes of seen classes.

2. In the rest of paper, we use the term "graph" to refer to the linked data structures such as social or biological networks, so as to avoid ambiguity with neural network terminology.

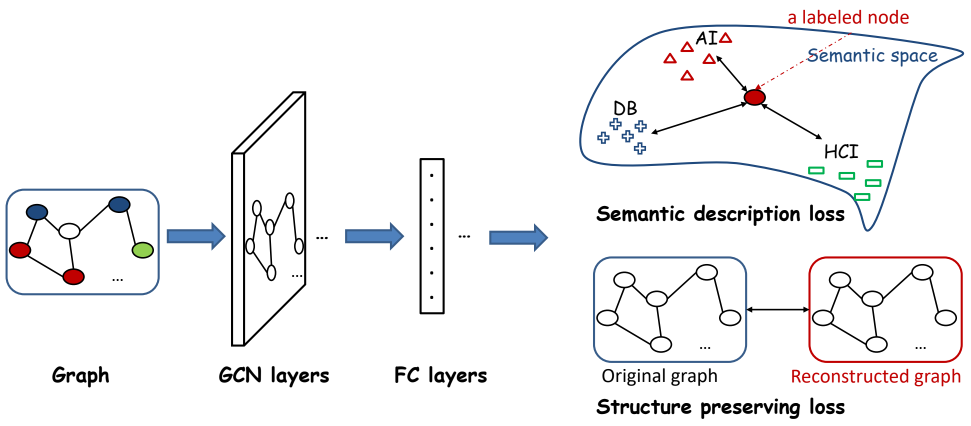


Fig. 4: Architecture overview of RECT.

4.2 Deduce Supervised Information for Unseen Classes

4.2.1 Observation

The recent success of ZSL demonstrates that the capacity of inferring semantic descriptions (also known as attributes) makes it possible to categorize unseen objects. Generally, the attributes are human annotated or provided by third-party resources (like the word embeddings learned from large-scale general text database), limiting the use of ZSL methods. In addition, the quality of attributes can be a source of problems in practical applications.

For graph embedding, we propose to obtain class-semantic descriptions in a more practical manner. To test this feasibility, we continue to use the citation graph Citeseer [26] as an example. Figure 3 shows some words sampled from the documents of AI, DB, and HCI classes in this dataset. Interestingly, these words also reflect some knowledge of other three (unseen) research areas (i.e., IR, ML and Agent). For example, IR's key words (like "human", "search" and "data") also show up in the documents of the (seen) research areas DB and HCI. This observation inspires us to generate class-semantic descriptions directly from the original node features.

4.2.2 Generate Class-semantic Descriptions Automatically

Let matrix $X \in \mathbb{R}^{n \times m}$ denote the feature matrix, where $x_i \in \mathbb{R}^m$ (the i-th row of X) is the corresponding m-dimensional feature vector of node v_i. To obtain the semantic descriptions for a seen class c, we can leverage a readout function $\mathcal{R}$, and use it to summarize a class-semantic description vector (denoted as $\hat{y}_c$) from the labeled nodes, i.e., $\hat{y}_c = \mathcal{R}(\{x_i | \forall_i \, \mathcal{C}_i^s = c\})$.

For those graphs without node features, we can treat the rows of adjacency matrix as node features. Intuitively, each node can be seen as a word, and all nodes construct a dictionary.

4.3 The Proposed Model: RECT

The architecture of RECT is illustrated in Fig. 4. In detail, we first adopt GCN layers to explore graph structure information. After propagating through all CGN layers, fully-connected (FC) layers are used to project the outputs of GCN layers into a semantic vector space, in which the loss is computed. Here, we use FC layers rather than GCN layers, because we hope to improve the robustness of the learned embeddings by satisfying our objective function without directly using the graph structure knowledge.

Our loss function consists of two parts. The first one is a prediction loss in the semantic space, i.e., the loss between the predicted and the actual class-semantic description vectors:

$$\mathcal{J}_{semantic} = \sum_{i \in \mathcal{L}} loss(\hat{y'}_{\mathcal{C}_i^s}, \hat{y}_{\mathcal{C}_i^s}) \tag{17}$$

where $\hat{y'}_{\mathcal{C}_i^s}$ and $\hat{y}_{\mathcal{C}_i^s}$ is the predicted and the actual class-semantic vector of the labeled node v_i respectively, and $loss(\cdot, \cdot)$ is a sample-wise loss function. By using this loss, our method can capture the class-semantic knowledge, making the learned graph embeddings reflect the supervised information of both seen and unseen classes.

The second is a graph structure preserving loss. Unlike GCN or other semi-supervised GNNs, we still propose to explicitly preserve the graph structure knowledge. This is because the above loss (Eq. 17) actually indirectly preserves the label discrimination, which would reduce the discrimination of learned embeddings (especially in the seen classes). For simplicity, here we follow the similar idea of our shallow method RSDNE. Specifically, the learned node embeddings U (i.e., the outputs of the last layer) should minimize:

$$\mathcal{J}_{graph\_nerual} = loss(M, UU') \tag{18}$$

To learn powerful embeddings by considering both parts, a simple and effective way we find in practice is to train the model which considers these two parts separately and then concatenate the embeddings trained by the two parts for each node. A more principled way to combine these two loss parts is to jointly train the objective functions Eq. 17 and Eq. 18, which we leave as future work.

For clarity, we summarize this method in Alg. 2. We refer this method as **RE**laxed G**C**N Ne**T**work (RECT), as it utilizes GCN model and relaxes the original label discrimination by preserving class-semantic knowledge.

Algorithm 2 RECT

```
Require: Graph information (i.e., $A$ and $X$), label information $\mathcal{L}$;
Ensure: The learned node embedding result $U$;
1: Summarize the class-semantic descriptions of seen classes through the readout function $\mathcal{R}$;
2: Obtain the embedding result $U^{(1)}$ by optimizing RECT with the objective function Eq. 17 ;
3: Obtain the embedding result $U^{(2)}$ by optimizing RECT with the objective function Eq. 18 ;
4: Obtain the final embedding result $U$ by concatenating the normalized $U^{(1)}$ and $U^{(2)}$ ;
5: return $U$.
```

5 Algorithm Analysis

5.1 Optimization Algorithm Solving Problem (14)

Theorem 1. *The optimal solution of problem (14) is Eq. 15.*

Proof. By contradiction, suppose a labeled node v_i has gotten its optimal intra-class neighbor set $\mathcal{N}_{kc}$ which contains a node v_p not in v_i's top-k nearest and same labeled nodes. As such, there must exist a node $v_q \notin \mathcal{N}_{kc}$ which is one of v_i's top-k nearest and same labeled nodes. Then, we get $\|u_i - u_p\|_F^2 > \|u_i - u_q\|_F^2$. Considering our minimization problem (i.e., Eq. 14), this inequation leads:

$$\sum_{j \in \mathcal{N}_{kc}} \|u_i - u_j\|_F^2 > \sum_{j \in \{\mathcal{N}_{kc}+v_q\} \setminus v_p} \|u_i - u_j\|_F^2 \tag{19}$$

This indicates that $\{\mathcal{N}_{kc}+v_q\}\setminus v_p$ is a better optimal solution than $\mathcal{N}_{kc}$, a contradiction. □

5.2 Time Complexity Analysis

Complexity of RSDNE: Following [49], the time complexity of Alg. 1 is as below. The complexity for updating U is $O(nnz(M)d + d^2n + nnz(L)d)$, where $nnz(\cdot)$ is the number of non-zeros of a matrix. The complexity for updating H is $O(nnz(M)d + d^2n)$. The complexity for updating S is $O(|\mathcal{C}^s|\ell^2 \log \ell)$, where $\ell = rn|\mathcal{C}^s|/|\mathcal{C}|$ is the average number of labeled nodes per class, and r is the label rate. As ℓ is linear with n and $nnz(L)$ is linear with $nnz(M)$, the overall complexity of RSDNE is $O(\tau(nnz(M)d+n^2 \log n))$, where τ is the number of iterations to converge.

Complexity of RSDNE*: For the light version, i.e., RSDNE*, the complexity of updating S becomes $O(|\mathcal{C}^s|\bar{k}^2 \log \bar{k})$, and all others remain the same. Hence, as $\bar{k} \ll n$, the overall complexity becomes $O(\tau(nnz(M)d+d^2n))$. As our method typically converges fast ($\tau \leq 15$ in our experiments) and $d \ll n$, the complexity of RSDNE* is linear to $nnz(M)$ and node number n.

Complexity of RECT: First of all, the time cost of the GCN layer is linear in the number of graph edges [20]. Specifically, the time complexity is $O(m|\mathcal{E}||d^h||\mathcal{C}^{stc}|)$, where $|\mathcal{E}|$ is the edge number and $|d^h|$ is the hidden layer dimension size and $|\mathcal{C}^{stc}|$ is the dimension of class-semantic description. The complexity of calculating Eq. 17 is $O(n|\mathcal{C}^{stc}|)$. The complexity of calculating Eq. 18 is $O(dn^2)$. Therefore, the total complexity of RECT is $O(m|\mathcal{E}||d^h||\mathcal{C}^{stc}|+n|\mathcal{C}^{stc}|+dn^2)$. Note we can directly reduce this complexity by adopting other graph structure preserving objectives, like the objective of DeepWalk (i.e., Eq. 1). Then, the total complexity will reduce to $O(m|\mathcal{E}||d^h||\mathcal{C}^{stc}| + n|\mathcal{C}^{stc}| + dn \log n)$, indicating the similar complexity as DeepWalk and GCN. We will test the efficient of this acceleration strategy in our experiments.

5.3 The Proposed Methods v.s. Traditional Semi-supervised Methods

5.3.1 Traditional Semi-supervised Methods

To benefit from the discriminative information (e.g., class labels), the most effective and widely used strategy is to guarantee both the intra-class similarity and inter-class dissimilarity in the embedding space [50], [51]. For this purpose, traditional semi-supervised graph embedding methods reduce the intra-class embedding variance and enlarge the inter-class embedding variance by optimizing various classification models. However, as the unseen class nodes are (partly) linked with the seen class ones (i.e., seen and unseen class nodes are correlated), only optimizing over the seen classes is suboptimal for the whole graph.

In those shallow methods (like LSHM), this suboptimal strategy would impose lots of strict constraints (like the "close-to" constraints between same labeled nodes) only on seen classes, which may seriously mislead the jointly trained graph structure preserving model and finally lead to very poor results. Similarly in those GNNs which implicitly preserve the graph structure, this suboptimal strategy would also mislead the used message aggregation mechanism and finally lead to very poor results.

5.3.2 The Relation of RSDNE

RSDNE actually relaxes these above-mentioned strict constraints in shallow methods. We show in the following that the intra-class similarity loss defined in [50] is a special case of our Eq. 5. This equivalence also explains the rationale of our method.

Theorem 2. *In each seen class c, let k_c and l_c denote the intra-class neighbor number and the labeled node number in this class, respectively. For each labeled class c, if we enlarge k_c to l_c, Eq. 5 is equivalent to the intra-class similarity equation.*

Proof. The intra-class similarity function in [50] is defined to minimize:

$$\sum_{i=1}^{n} \sum_{j:\mathcal{C}_i^s=\mathcal{C}_j^s} \|u_i - u_j\|_F^2 \tag{20}$$

In each seen class c, if we set $k_c = l_c$, Eq. 5 actually minimizes:

$$\begin{aligned} &\sum_{i,j=1}^{n} \|u_i - u_j\|_F^2 S_{ij} \\ \text{s.t. } &\forall i,j \in \mathcal{L}, S_{ij} \in \{0,1\}, \text{ if } \mathcal{C}_i^s = \mathcal{C}_j^s \\ &\forall i,j, S_{ij} = 0, \text{ if } i \notin \mathcal{L} \text{ or } \mathcal{C}_i^s \neq \mathcal{C}_j^s \end{aligned} \tag{21}$$

As Eq. 21 equals Eq. 20, the conclusion is proved. □

Similarly, in RSDNE, the objective function part formulated in Eq. 6 actually relaxes the classical inter-class dissimilarity. Specifically, in Eq. 6, W_{ij} measures the similarity score between node v_i and v_j. For two different labeled nodes v_i and v_j, setting W_{ij} to a large negative number

TABLE 2. The Statistics of Datasets.

Name	Citeseer	Cora	Wiki	PPI	Blogcatalog
Type	Citation graph	Citation graph	Hyperlink graph	Biological graph	Social graph
Nodes	3,312	2,708	2,405	3,890	10,312
Edges	4,732	5,429	17,981	76,584	333,983
Classes	6	7	17	50	39
Features	3,703	1,433	4,973	-	-
Multi-label	No	No	No	YES	YES

reflects the intuition of inter-class dissimilarity. In sum, these two relaxation strategies not only reasonably guarantee both intra-class similarity and inter-class dissimilarity, but also avoid misleading the jointly trained graph structure preserving model. Consequently, RSDNE would benefit from completely-imbalanced labels, which is further verified in our experiments.

5.3.3 The Relation of RECT

RECT and traditional GNNs share the similar neural network architecture. The fundamental difference is the objective function. Traditional GNNs preserve the class-label discrimination. RECT aims to preserve the class-semantic knowledge. As shown in related ZSL studies, class-semantic knowledge enables the knowledge transfer from seen classes to unseen classes, making the learned embeddings reflect the supervised knowledge of both seen and unseen classes. Intuitively, RECT can also be seen as a relaxation of the class-label discrimination by preserving the class-semantic knowledge.

6 Experiments

Datasets: We conduct our experiments on five real-world graphs, whose statistics are listed in Table 2. Citeseer [26] and Cora [26] are citation graphs whose nodes are articles, edges are citations, and labels are research areas. Wiki [52] is a set of Wikipedia pages. In this dataset, nodes are web pages, edges are hyperlinks among them, and labels are topics. PPI [7] is a biological graph dataset, and BlogCatalog [53] is a social graph dataset. Their labels are biological states and user interests, respectively. Unlike the previous ones, the nodes in these two graphs may have multiple labels. In addition, these two graphs do not have node features, and we use the rows of their adjacency matrices as node features.

Baseline Methods: We compare the proposed methods against the following baselines:

1) NodeFeats is a content-only baseline which only uses the original node features.
2) MFDW [44] is the matrix factorization form of DeepWalk [6]. This method is unsupervised.
3) LINE [13] is also a popular unsupervised method which considers the first-order and second-order proximity information.
4) LSHM [17] is a semi-supervised method which considers the first-order proximity of a graph and jointly learns a linear classification model.
5) LDE [18] is a semi-supervised method which also considers the first-order proximity and jointly trains a 1-nearest neighbor classification model.
6) MMDW [19] is a semi-supervised method which adopts MFDW model to preserve the graph structure and jointly trains an SVM model.
7) TADW [44] is a unsupervised method which incorporates DeepWalk and associated node features into the matrix factorization framework.
8) DGI [54] is a recently proposed unsupervised GNN method which trains a graph convolutional encoder through maximizing mutual information.
9) GCN [20] is the most well-known GNN method. This method is supervised.
10) APPNP [22] extends GCN with the idea of PageRank to explore the global graph structure. This method is also supervised.

Parameters: Following [19], the embedding dimension is set to 200. In addition, for DeepWalk, we adopt the default parameter setting i.e., window size is 5, walks per vertex is 80. For LINE, we first learn two 100-dimension embeddings by adopting its first-order proximity and second-order proximity separately, and then concatenate them as suggested in [13]. To fully show the limitations of those semi-supervised methods, we also tune their parameters by a grid-search strategy from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$ and report the best results. For these three GNNs (DGI, GCN and APPNP), we all use the code provided by the authors and adopt the default hyper-parameters. As GCN and APPNP are end-to-end node classification methods, we use the outputs of their hidden layer (whose hidden units number is set to 200) as embedding results. Additionally, as the original implementations of GCN and APPNP do not support multi-label tasks, we replace their loss functions by Binary Cross-entropy loss on PPI and Blogcatalog datasets as [55].

In contrast, in RSDNE and its light version RSDNE*, we fix parameters α=1 and λ=0.1 throughout the experiment. In addition, we simply set the intra-class neighbor number k=5 like most manifold learning methods [56], and set the candidate number $\bar{k}$=$20k$ for RSDNE*.

The settings of our RECT method and its two sub-methods are as follows. We use RECT-L to denote the sub-method with the semantic preserving loss (i.e., Eq. 17), and we use RECT-N to denote the sub-method with the graph preserving loss (i.e., Eq. 18). In RECT-L, we train a simple model with one GCN layer and one FC layer. In addition, we use a simple averaging function as its readout function $\mathcal{R}$; and we apply SVD decomposition on the original node features to get 200-dimensional node features, for the calculation of semantic preserving loss. In RECT-N, we train a simple model with only one GCN layer. In both sub-methods, we use the PReLU activation [57], mean squared error (MSE) loss, and Xavier initialization [48]. We train

TABLE 3. Micro-F1 scores on classification tasks. The best result is marked in bold. In the case of no node features, the best result is marked with underline.

Information		X	A		$A,\mathcal{L}$			A,X		$A,X,\mathcal{L}$		$A,\mathcal{L}$		A,X	$A,X,\mathcal{L}$	
Data \ *Method*		NodeFeats	MFDW	LINE	LSHM	LDE	MMDW	TADW	DGI	GCN	APPNP	RSDNE	RSDNE*	RECT-N	RECT-L	RECT
Citeseer	10%	0.6535	0.4810	0.4448	0.4253	0.4515	0.5141	0.6844	0.7014	0.5640	0.5944	0.5395	0.5426	0.6975	0.6601	**0.7083**
	30%	0.7006	0.5793	0.4959	0.5504	0.5224	0.6020	0.7187	0.7293	0.5889	0.6274	0.6313	0.6271	0.7301	0.7154	**0.7403**
	50%	0.7161	0.6096	0.5084	0.6027	0.5805	0.6278	0.7276	0.7377	0.5995	0.6356	0.6741	0.6683	0.7359	0.7294	**0.7475**
Cora	10%	0.6508	0.6699	0.6678	0.5981	0.6641	0.7149	0.7978	0.7996	0.6436	0.7068	0.7569	0.7513	0.8187	0.7617	**0.8197**
	30%	0.7214	0.7908	0.7220	0.7254	0.7449	0.7939	0.8245	0.8350	0.6696	0.7347	0.8184	0.8147	0.8524	0.8208	**0.8561**
	50%	0.7589	0.8164	0.7373	0.7487	0.7705	0.8135	0.8361	0.8366	0.6786	0.7607	0.8426	0.8372	0.8550	0.8331	**0.8615**
Wiki	10%	0.1741	0.3570	0.5586	0.4319	0.4920	0.5582	0.5899	0.5423	0.6616	0.6189	0.5803	0.5822	0.7028	0.7006	**0.7180**
	30%	0.2212	0.5579	0.6170	0.5658	0.5846	0.6224	0.6669	0.6005	0.6952	0.6463	0.6477	0.6493	0.7363	0.7534	**0.7580**
	50%	0.2616	0.6303	0.6434	0.5838	0.6158	0.6419	0.6845	0.6274	0.7033	0.6578	0.6772	0.6751	0.7457	0.7704	**0.7711**
PPI	10%	0.0980	0.1447	0.1391	0.0306	-	-	0.1379	0.1433	0.0469	0.0439	-	-	0.1518	0.1537	**0.1659**
	30%	0.1390	0.1799	0.1693	0.0626	-	-	0.1724	0.1671	0.0449	0.0458	-	-	0.1873	0.1773	**0.1956**
	50%	0.1660	0.1833	0.1816	0.0891	-	-	0.1809	0.1715	0.0438	0.0410	-	-	0.1960	0.1834	**0.2065**
Blogcatalog	10%	0.2683	0.3192	0.3311	0.1632	-	-	0.3302	0.2371	0.0271	0.1121	-	-	0.3372	0.3076	**0.3399**
	30%	0.2984	0.3436	0.3504	0.2357	-	-	0.3409	0.2654	0.0316	0.1364	-	-	0.3571	0.3261	**0.3627**
	50%	0.3249	0.3485	0.3600	0.2803	-	-	0.3431	0.2741	0.0492	0.1365	-	-	0.3621	0.3321	**0.3692**

TABLE 4. Macro-F1 scores on classification tasks. The bold mark and underline mark have the same meanings as in Table. 3.

Information		X	A		$A,\mathcal{L}$			A,X		$A,X,\mathcal{L}$		$A,\mathcal{L}$		A,X	$A,X,\mathcal{L}$	
Data \ *Method*		NodeFeats	MFDW	LINE	LSHM	LDE	MMDW	TADW	DGI	GCN	APPNP	RSDNE	RSDNE*	RECT-N	RECT-L	RECT
Citeseer	10%	0.5860	0.4195	0.3856	0.3724	0.4155	0.4707	0.6294	0.6310	0.4761	0.5175	0.4949	0.4994	0.6233	0.6089	**0.6541**
	30%	0.6504	0.5253	0.4315	0.4990	0.5030	0.5606	0.6679	0.6382	0.5085	0.5567	0.5939	0.5852	0.6669	0.6647	**0.6919**
	50%	0.6692	0.5559	0.4403	0.5454	0.5540	0.5835	0.6788	0.6595	0.5159	0.5665	0.6385	0.6285	0.6798	0.6799	**0.7016**
Cora	10%	0.6182	0.6598	0.6478	0.5595	0.6453	0.7043	0.7823	0.7540	0.5623	0.6308	0.7436	0.7367	0.8084	0.7457	**0.8094**
	30%	0.7103	0.7819	0.7099	0.6625	0.7343	0.7830	0.8127	0.8257	0.5856	0.6641	0.8073	0.8029	0.8438	0.8008	**0.8462**
	50%	0.7430	0.8081	0.7284	0.6798	0.7628	0.8045	0.8245	0.8277	0.5991	0.6937	0.8318	0.8267	0.8455	0.8196	**0.8502**
Wiki	10%	0.0538	0.2835	0.4025	0.3099	0.3872	0.4190	0.4538	0.3615	0.4680	0.4031	0.4518	0.4468	0.5405	0.5525	**0.5789**
	30%	0.1110	0.4333	0.4738	0.3869	0.4641	0.4973	0.5651	0.4270	0.4939	0.4365	0.5326	0.5363	0.6093	0.6206	**0.6480**
	50%	0.1530	0.4958	0.5136	0.4209	0.5047	0.5257	0.6208	0.4387	0.4954	0.4486	0.5741	0.5655	0.6340	0.6490	**0.6573**
PPI	10%	0.0574	0.0915	0.0854	0.0148	-	-	0.0851	0.0833	0.0153	0.0156	-	-	0.0966	0.1133	**0.1191**
	30%	0.0902	0.1204	0.1040	0.0316	-	-	0.1102	0.0980	0.0156	0.0189	-	-	0.1262	0.1238	**0.1402**
	50%	0.1083	0.1205	0.1222	0.0522	-	-	0.1183	0.1070	0.0141	0.0176	-	-	0.1327	0.1248	**0.1491**
Blogcatalog	10%	0.1008	0.1488	0.1472	0.0385	-	-	0.1438	0.0794	0.0131	0.0281	-	-	0.1596	0.1187	**0.1622**
	30%	0.1157	0.1721	0.1727	0.0894	-	-	0.1571	0.1042	0.0139	0.0299	-	-	0.1887	0.1335	**0.1921**
	50%	0.1369	0.1787	0.1806	0.1285	-	-	0.1584	0.1166	0.0151	0.0293	-	-	0.1974	0.1396	**0.1997**

all models for 100 epochs (training iterations) using Adam SGD optimizer [58] with a learning rate of 0.001. Unless otherwise noted, all these settings are tested throughout the experiments.

6.1 Test with Completely-imbalanced Label

Experimental setting: Following [6], we validate the quality of learned representations on node classification task. As this study focuses on the completely-imbalanced label setting, we need to perform seen/unseen class split and remove the unseen classes from the training data. Particularly, for Citeseer and Cora, we use two classes as unseen. Thus, we have C_6^2 and C_7^2 different seen/unseen splits for Citeseer and Cora, respectively. As Wiki, PPI and Blogcatalog contain much more classes, we randomly select five classes as unseen classes and repeat the split for 20 times.

The detailed experimental procedure is as follows. First, we randomly sample some nodes as the training set (denoted as $\mathcal{L}$), and use the rest as the test set. Then, we remove the unseen class nodes from $\mathcal{L}$ so as to obtain the completely-imbalanced labeled data $\mathcal{L}'$. With the graph knowledge (i.e., A and X) and $\mathcal{L}'$, we get the representations learned by various methods. Note that no methods can use the labeled data from unseen classes for embedding. After that, we train a linear SVM classifier based on the learned representations and the original label information $\mathcal{L}$. At last, the trained SVM classifier is evaluated on the test data.

6.1.1 Node Classification Performance

We vary the percentage of labeled data in [10%, 30%, 50%] and then use the labeled nodes of seen classes as supervision for graph embedding learning. We employ two widely used classification evaluation metrics: Micro-F1 and Macro-F1 [59]. In particular, Micro-F1 is a weighted average of F1-scores over different classes, while Macro-F1 is an arithmetic mean of F1-scores on each label:

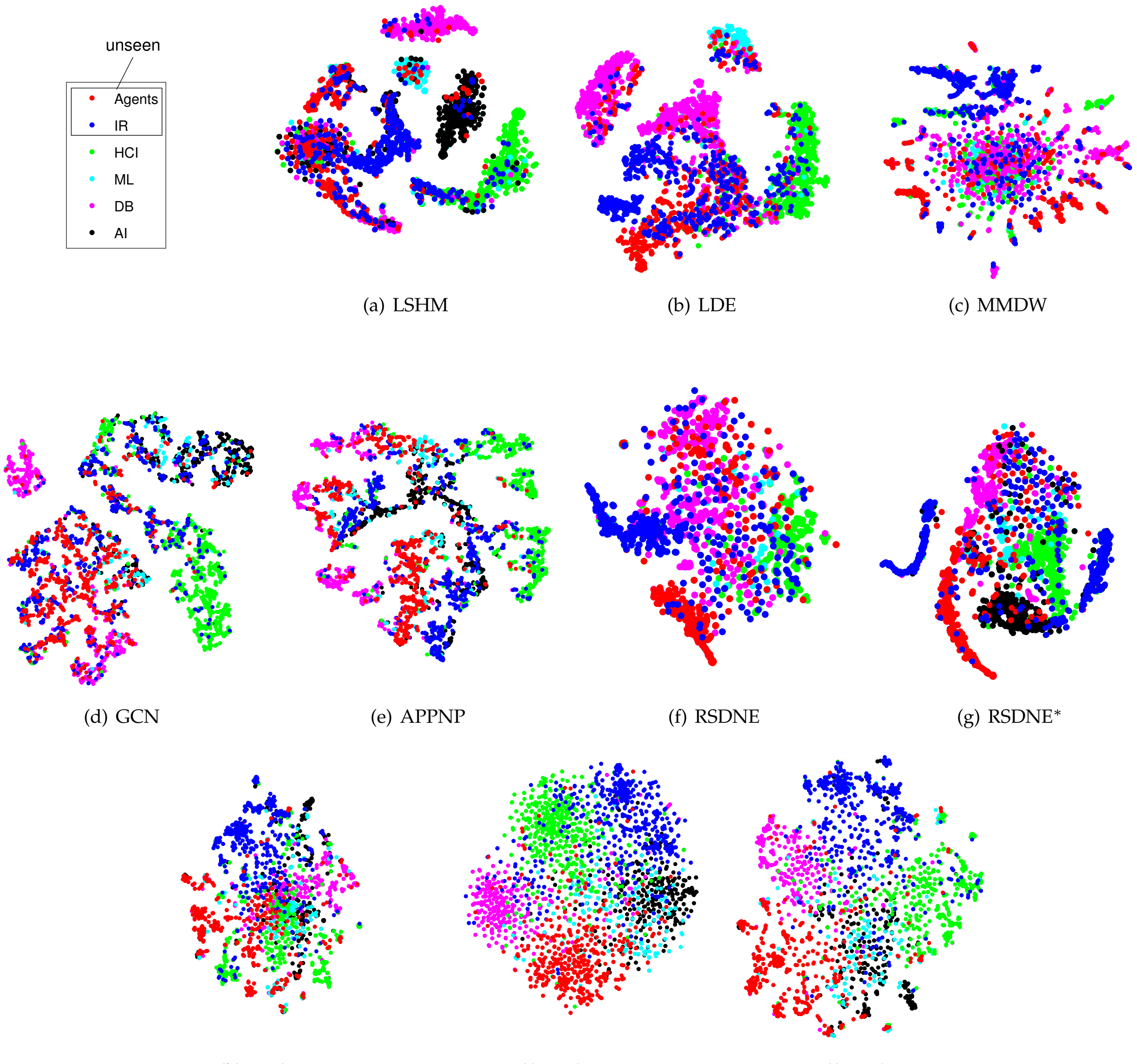
$$\begin{aligned} \text{Micro}-\text{F1} &= \frac{\sum_{i=1}^{|\mathcal{C}|} 2TP^i}{\sum_{i=1}^{|\mathcal{C}|}(2TP^i + FP^i + FN^i)} \\ \text{Macro}-\text{F1} &= \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|} \frac{2TP^i}{(2TP^i + FP^i + FN^i)} \end{aligned} \tag{22}$$

where $|\mathcal{C}|$ is the class number, TP^i denotes the number of positives in the i-th class, FP^i and FN^i denotes the number of false positives and false negatives in the i-th class, respectively.

The results are presented in Tables 3 and 4, from which we have the following observations [3].

Firstly, our deep method RECT always achieves the best results on all datasets including both single-label and multi-label graphs. This can be explained by the performance of RECT-L. We can clearly find that RECT-L always outperforms the compared semi-supervised GNNs (i.e., GCN

3. We do not test LDE, MMDW, RSDNE, and RSDNE* on PPI and Blogcatelog, since these methods could not handle the multi-label case.



(a) LSHM (b) LDE (c) MMDW

(d) GCN (e) APPNP (f) RSDNE (g) RSDNE*

(h) RECT-N (i) RECT-L (j) RECT

Fig. 5: 2D visualization on Citeseer (50% label rate with two unseen classes, i.e.,{Agents, IR}).

and APPNP) by a large margin (around 20% to 300% relatively). This indicates that, by exploring the class-semantic knowledge, RECT can effectively utilize the completely-imbalanced labels.

Secondly, our shallow method RSDNE and its light version both perform much better than all baselines which do not use node attributes. For example, with 50% labeled data, our two methods outperform the best baseline MMDW by 7–12% relatively in term of Micro-F1. The underlying principle is that our approximation models (i.e., Eq. 5 and Eq. 6) reasonably guarantee both intra-class similarity and inter-class dissimilarity, and meanwhile avoids misleading the jointly trained graph structure preserving model. Besides, the light version of our method RSDNE* is competitive with RSDNE. This means that we can reduce the intra-class neighbor candidate number to make our method more efficient.

Thirdly, our deep method RECT is more powerful than our shallow method RSDNE. For example, in Citeseer with 30% labeled data, RECT outperforms RSDNE by 17.46% relatively in term of Micro-F1. The reason mainly lies in two folds. On the one hand, benefiting from the powerful GNN layers, RECT could utilize the attributes of nodes. On the other hand, exploring the knowledge of class-semantic descriptions (via a simple readout function) enables RECT to handle multi-label setting.

Lastly, all compared semi-supervised baselines become ineffective, and some of them even perform worse than unsupervised ones. For example, LSHM and LDE achieve lower accuracy than MFDW in most cases; GCN and APPNP also perform worse than DGI almost all the time.

This is consistent with our theoretical analysis (Section 5.3) that traditional semi-supervised methods could get unappealing results in this completely-imbalanced label setting.

6.1.2 Graph Layouts

Following [13], we use t-SNE package [60] to map the learned representations of Citeseer into a 2D space. Without loss of generality, we simply adopt Citeseer's first two classes as unseen classes, and set the training rate to 50%. (Due to space limitation, we only visualize the embeddings obtained by semi-supervised methods.)

First of all, the visualizations of our GNN method (RECT and its sub-methods), as expected, exhibit the most discernible clustering. Especially, as shown in Fig. 5(i), RECT-L which utilizes label information successfully respects the six topic classes of Citeseer. In this visualization, we also note that the clusters of different classes do not separate each other by a large margin. This is consistent with our analysis that the class-semantic preservation can be seen a relaxation of the classical classification loss. Additionally, as shown in Fig. 5(j), RECT obtains the best visualization result, in which different topic classes are clearly separated.

Additionally, the visualizations of our RSDNE and RSDNE* are also quite clear, with meaningful layout for both seen and unseen classes. As shown in Figs. 5(f-g), the nodes of the same class tend to lie on or close to the same manifold. Notably, the nodes from two unseen classes avoid heavily mixing with the wrong nodes. Another surprising observation is that: compared to RSDNE, the embedding results of its light version (i.e., RSDNE*) seem to lie on more compact manifolds. The reason might be that RSDNE* has a stricter manifold constraint, i.e., a labeled node's k intra-class neighbors are adaptively selected from a predetermined candidate set. The similar observation can be found in traditional manifold learning methods [12] in which the neighbor relationships among instances are predetermined.

In contrast, all the compared semi-supervised baselines get unappealing visualizations. For example, as shown in Figs. 5(a-b), although LSHM and LDE better cluster and separate the nodes from different seen classes, their two kinds of unseen class nodes heavily mix together. The similar observation can be found in the results of semi-supervised GNNs (i.e., GCN and APPNP), as shown in Figs. 5(d-e). In addition, as shown in Fig. 5(c), MMDW also fails to benefit from the completely-imbalanced labels. This is because MMDW has to use a very small weight for its classification model part to avoid poor performance.

6.1.3 Effectiveness Verification

In the following experiments, we only show the results on Citesser, since we get similar results on the other datasets.

Effect of Seen/Unseen Class Number: Without loss of generality, we set the training rate to 50%, and vary the seen class number from six to one on Citeseer. As shown in Fig. 6, RSDNE and RECT can constantly benefit from the completely-imbalanced labels. For example, even with only one seen class, RSDNE still outperforms (its unsupervised version) MFDW; RECT still outperforms (its unsupervised version) RECT-N. Besides, the performance of RECT-L declines smoothly when the unseen class number grows, clearly demonstrating the effectiveness of exploring the class-semantic knowledge for the studied problem.

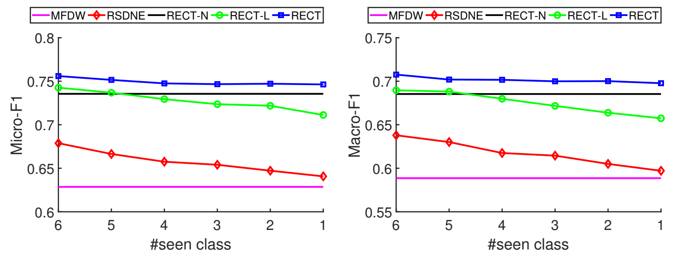


Fig. 6: Node classification performance w.r.t. the seen class number on Citeseer (with 50% label rate).

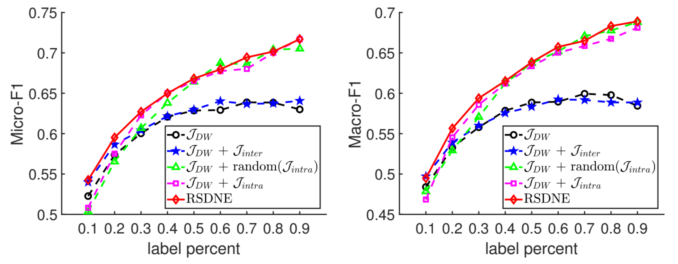


Fig. 7: Node classification performance w.r.t. different settings of RSDNE on Citeseer.

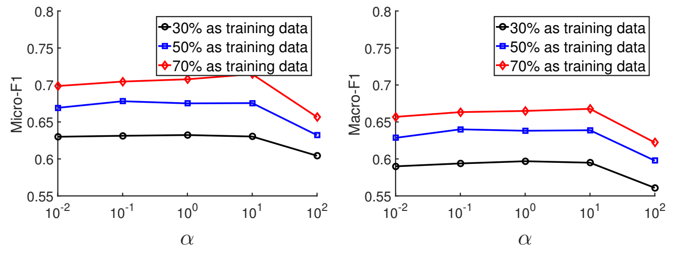


Fig. 8: The effect of parameter α in RSDNE on Citeseer.

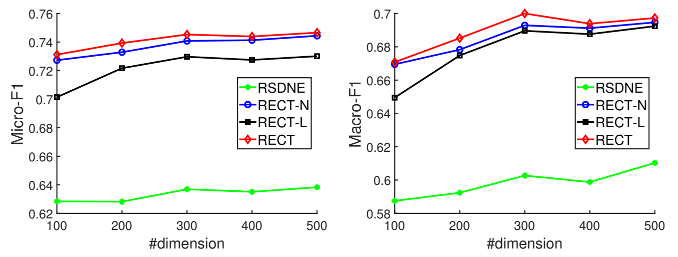


Fig. 9: The effect of embedding dimension on Citeseer with label rate 30%.

Effect of Intra-class Similarity and Inter-class Dissimilarity Modeling in RSDNE: To investigate the effect of these two parts, we test the following settings of RSDNE:

1) $\mathcal{J}_{DW}$: only modeling the graph structure (Eq. 4).
2) $\mathcal{J}_{DW}+\mathcal{J}_{intra}$: modeling graph structure and intra-class similarity (selecting intra-class neighbors adaptively (Eq. 5)).

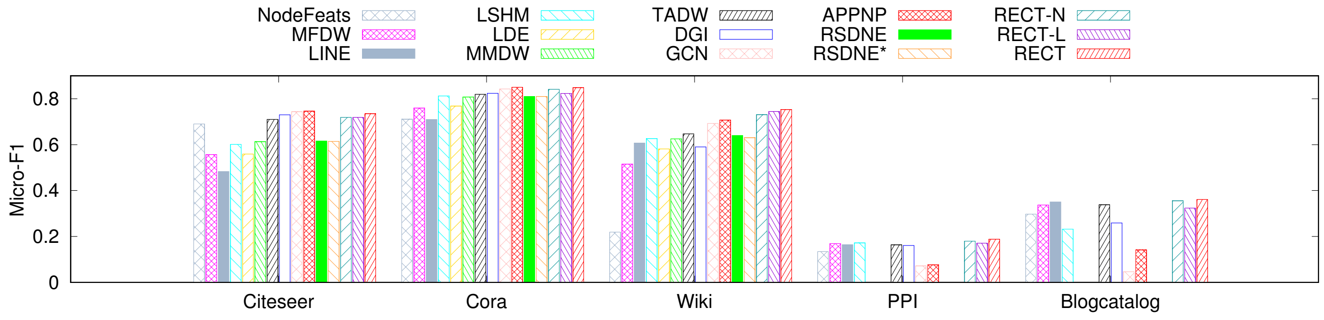


Fig. 10: Averaged node classification performance (Micro-F1) with balanced labels.

3) $\mathcal{J}_{DW}$+ random($\mathcal{J}_{intra}$): modeling graph structure and intra-class similarity (selecting intra-class neighbors randomly).
4) $\mathcal{J}_{DW}$+$\mathcal{J}_{inter}$: modeling graph structure and inter-class dissimilarity (Eq. 6).

As shown in Fig. 7, when either eliminating the effect of intra-class or inter-class modeling part, the performance degrades. This suggests that these two parts contain complementary information to each other for graph embedding. Another interesting observation is that: although randomly selecting intra-class neighbors (i.e., $\mathcal{J}_{DW}$+ random($\mathcal{J}_{intra}$)) does not show the best result, it still outperforms modeling graph structure alone (i.e., $\mathcal{J}_{DW}$) significantly, especially when the labeled data set becomes larger. This again shows the effectiveness of modeling the (relaxed) intra-class similarity.

6.1.4 *Sensitivity Analysis*

Sensitivity of Parameter: In the proposed method RSDNE, there is an important parameter α which balances the contributions of graph structure and label information. Figure 8 shows the classification performance with respect to this parameter on Citeseer (with the regularization parameter λ=0.1). It can be observed that our method is not sensitive to α especially when $\alpha \in [10^{-2}, ..., 10^{1}]$.

Sensitivity of Embedding Dimension: We vary embedding dimensions in {100, 200, 300, 400, 500}. As shown in Fig. 9, all our methods are not very sensitive to the embedding dimension. In addition, we can find that RECT always outperforms its two sub-methods RECT-N and RECT-L. Another observation needs to be noted is that RECT still outperforms all baselines when the embedding dimension is set to 200. All these observations demonstrate the superiority of our methods.

6.2 Test with Balanced Labels

We also test the suitation where the labeled data is generally balanced, i.e., the labeled data covers all classes. Figure 10 shows the averaged classification performance (training ratio also varies in [10%, 30%, 50%]). We can get the following two interesting observations.

The first and the most interesting observation is that our methods have comparable performance to state-of-the-art semi-supervised methods, although our methods are not specially designed for this balanced case. Specifically, RSDNE and RSDNE* obtain comparable performance to LSHM, LDE and MMDW; RECT obtains comparable (and sometimes much superior) results to GCN and APPNP. This suggests that our methods would be favorably demanded by the scenario where the quality of the labeled data cannot be guaranteed.

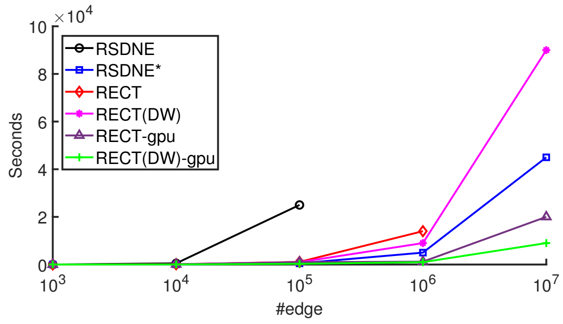


Fig. 11: Training time of our methods. We do not report the running time when it exceeds 25 hours.

The second observation is that our deep method RECT is more robust than the compared deep semi-supervised GNNs. As shown in Fig. 10, GCN and APPNP perform poorly on two multi-label datasets PPI and Blogcatalog. This may due to the imbalanced of labels in these two datasets. In contrast, our method RECT is much more stable on all datasets. This might indicate that the distribution of class-semantic descriptions over various classes is more balanced than that of class labels. All these observations show the general applicability of our approximation models (i.e., Eq. 5, Eq. 6 and Eq. 17) which could also be considered in other related applications.

6.3 Scalability Test

Following [20], we use random graphs to test the scalability. Specifically, we create a random graph with n nodes and $2n$ edges. We take the identity matrix as the input feature matrix X. We give same label for all nodes, set training rate to 10% and do not remove any labeled nodes.

We test RSDNE, RSDNE* and together with different implements of our RECT method: 1) RECT is the original proposed GNN method; 2) RECT(DW) adopts the objective of DeepWalk for graph structure preserving (i.e., Eq. 1); 3) RECT-gpu is the GPU implementation of RECT; 4) RECT(DW)-gpu is the GPU implementation of RECT(DW). Our methods are written in Python 3.0 and Pytorch 1.0. All the codes are running on a server with 16 CPU cores, 32 GB main memory, and an Nvidia Titan V GPU. Figure 11 shows the running times. We can find that RSDNE* is more efficient than RSDNE, which is consistent with our theoretical analysis. We also find that RECT(DW) is more efficient than RECT, indicating we can adopt various graph structure preserving objectives to accelerate our method. In addition, the GPU implementation of GNN methods can largely accelerate the training speed.

7 Conclusion

This paper investigates the graph embedding problem in the completely-imbalanced label setting where the labeled data cannot cover all classes. We firstly propose a shallow method named RSDNE. Specifically, to benefit from completely-imbalanced labels, RSDNE guarantees both intra-class similarity and inter-class dissimilarity in an approximate way. Then, to leverage the power of deep neural networks, we propose RECT, a new class of GNN. Unlike RSDNE, RECT utilizes completely-imbalanced labels by exploring the class-semantic descriptions, which enables it to handle graphs with node features and multi-label setting. Finally, extensive experiments are conducted on several real-world datasets to demonstrate the effectiveness of the proposed methods.

References

[1] Z. Wang, C. Wang, J. Pei, X. Ye, and S. Y. Philip, "Causality based propagation history ranking in social networks." in *International Joint Conference on Artificial Intelligence*, 2016, pp. 3917–3923.

[2] C. Wang, H. Wang, C. Zhou, J. Li, and H. Gao, "Ecoqug: An effective ensemble community scoring function," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1702–1705.

[3] H. Wang, N. Li, J. Li, and H. Gao, "Parallel algorithms for flexible pattern matching on big graphs," *Information Sciences*, vol. 436, pp. 418–440, 2018.

[4] W. Li, M. Qiao, L. Qin, Y. Zhang, L. Chang, and X. Lin, "Scaling distance labeling on small-world networks," in *Proceedings of the 2019 International Conference on Management of Data*. ACM, 2019, pp. 1060–1077.

[5] P. Bühlmann and S. Van De Geer, *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 701–710.

[7] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864.

[8] C. Wang, C. Wang, Z. Wang, X. Ye, J. X. Yu, and B. Wang, "Deepdirect: Learning directions of social ties with edge-based network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2277–2291, 2018.

[9] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 287–297.

[10] L. G. Moyano, "Learning network representations," *The European Physical Journal Special Topics*, vol. 226, no. 3, pp. 499–518, 2017.

[11] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[12] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[13] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[14] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1225–1234.

[15] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *IEEE International Joint Conference on Neural Networks*, 2005, pp. 729–734.

[16] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[17] Y. Jacob, L. Denoyer, and P. Gallinari, "Learning latent representations of nodes for classifying in heterogeneous social networks," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. ACM, 2014, pp. 373–382.

[18] S. Wang, J. Tang, C. Aggarwal, and H. Liu, "Linked document embedding for classification," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 115–124.

[19] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: discriminative learning of network representation," in *International Joint Conference on Artificial Intelligence*, 2016, pp. 3889–3895.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[22] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations*, 2019.

[23] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[24] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, 2005.

[25] G. de Melo, "Inducing conceptual embedding spaces from wikipedia," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 43–50.

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

[27] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[28] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[29] F. Herrera, F. Charte, A. J. Rivera, and M. J. Del Jesus, "Multilabel classification," in *Multilabel Classification*. Springer, 2016, pp. 17–31.

[30] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2251–2265.

[31] B. V. Dasarathy, "Nearest neighbor (nn) norms: Nn pattern classification techniques," *Los Alamitos: IEEE Computer Society Press, 1990*, 1990.

[32] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[33] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[34] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Transactions on Industrial Informatics*, 2019.

[35] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, 2018.

[36] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.

[37] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.

[38] Y. Yan, T. Yang, Y. Yang, and J. Chen, "A framework of online learning with imbalanced streaming data," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2016, pp. 2817–2823.

[39] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[40] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 951–958.

[41] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 1778–1785.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[43] C. Geng, S.-j. Huang, and S. Chen, "Recent advances in open set recognition: A survey," *arXiv preprint arXiv:1811.08581*, 2018.

[44] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *International Joint Conference on Artificial Intelligence*, 2015, pp. 2111–2117.

[45] L. Polok, V. Ila, and P. Smrz, "Fast radix sort for sparse linear algebra on gpu," in *Proceedings of the High Performance Computing Symposium*. Society for Computer Simulation International, 2014, p. 11.

[46] M. Belkin and P. Niyogi, "Convergence of laplacian eigenmaps," in *Advances in Neural Information Processing Systems*, 2007, pp. 129–136.

[47] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific Belmont, 1999.

[48] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[49] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Advances in Neural Information Processing Systems*, 2015, pp. 2107–2115.

[50] D. Lin and X. Tang, "Inter-modality face recognition," *European Conference on Computer Vision*, pp. 13–26, 2006.

[51] M. Kan, S. Shan, H. Zhang, S. Lao, and X. Chen, "Multi-view discriminant analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 188–194, 2016.

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

[53] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 817–826.

[54] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *International Conference on Learning Representations*, 2019.

[55] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu, "Disentangled graph convolutional networks," in *International Conference on Machine Learning*, 2019, pp. 4212–4221.

[56] X. Zhu, "Semi-supervised learning literature survey," *Computer Science, University of Wisconsin-Madison*, vol. 2, no. 3, p. 4, 2006.

[57] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[59] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, no. 1, pp. 69–90, 1999.

[60] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

Zheng Wang received the B.E. degree in computer science from Beijing Jiaotong University, in 2009 and the Ph.D. degree in Software Engineering from Tsinghua University, in 2018. Currently, he is an assistant professor at Department of Computer Science and Technology, University of Science and Technology Beijing, China. His research interests include social network analysis and artificial intelligence.

Xiaojun Ye received the BS degree in mechanical engineering from Northwest Polytechnical University, Xian, China, in 1987 and the Ph.D. degree in information engineering from INSA Lyon, France, in 1994. Currently, he is a professor at School of Software, Tsinghua University, Beijing, China. His research interests include cloud data management, data security and privacy, and database system testing.

Chaokun Wang received the B.Eng. degree in computer science and technology, the M.Sc. degree in computational mathematics and the Ph.D. degree in computer software and theory in 1997, 2000 and 2005, respectively, from Harbin Institute of Technology, China. He joined the faculty of School of Software at Tsinghua University in February 2006, where currently he is an Associate Professor. He has published over 60 refereed papers, got two best paper awards at international conferences and holds twelve patents. His current research interests include social network analysis, graph data management, and music computing.

Jian Cui is a master student in the School of Computer and Communication Engineering, University of Science and Technology Beijing, China. His research interests include social computing and artificial intelligence.

Philip S. Yu is a Distinguished Professor in Computer Science at the University of Illinois at Chicago and also holds the Wexler Chair in Information Technology. His research interest is on big data, including data mining, data stream, database and privacy. He has published more than 920 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. Dr. Yu is a Fellow of the ACM and of the IEEE. He is associate editors of ACM Transactions on the Internet Technology and ACM Transactions on Knowledge Discovery from Data. He is on the steering committee of IEEE Conference on Data Mining and was a member of the IEEE Data Engineering steering committee. Dr. Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University.