Data Science and Engineering (2018) 3:68–83
https://doi.org/10.1007/s41019-018-0062-8

Using Node Identifiers and Community Prior for Graph-Based Classification

Qi Ye[1] · Changlei Zhu[1] · Gang Li[1] · Zhimin Liu[1] · Feng Wang[1]

Received: 25 August 2017 / Revised: 8 November 2017 / Accepted: 6 March 2018 / Published online: 16 March 2018

Abstract
With widely available large-scale network data, one hot topic is how to adopt traditional classification algorithms to predict the most probable labels of nodes in a partially labeled network. In this article, we propose a new algorithm called identifier-based relational neighbor classifier (IDRN) to solve the within-network multi-label classification problem. We use the node identifiers in the egocentric networks as features and propose a within-network classification model by incorporating community structure information to predict the most probable classes for unlabeled nodes. We demonstrate the effectiveness of our approach on several publicly available datasets. First, taking a semi-supervised approach, IDRN without any community prior is applied in community detection experiments, and it outperforms most existing unsupervised community detection algorithms. After that, in large-scale graph-based multi-label classification tasks, our approaches perform well in both fully labeled and partially labeled networks in most cases. To evaluate the scalability of our algorithm, we also show a scalability test to evaluate the running time of our algorithm in different networks. The experiment results show that our approach is quite efficient and suitable for large-scale real-world classification tasks.

Keywords Within-network classification · Node classification · Collective classification · Relational learning

1 Introduction

Massive networks exist in various real-world applications. These networks may be only partially labeled due to manual labeling can be highly cost in real-world tasks. A critical problem is how to use the network structure and other extra information to build better classifiers to predict labels for the unlabeled nodes. Recently, much attention has been paid to this problem, and various prediction algorithms over nodes have been proposed [1–3].

In this article, we propose a within-network classifier by making use of the node identifiers in the egocentric networks as features and the community prior. Traditional relational classification algorithms, such as WvRN [4] and SCRN [5] classifier, make statistical estimations of the labels through statistics, class label propagation or relaxation labeling. From a different viewpoint, many real-world networks display some useful phenomena, such as clustering phenomenon [6] and scale-free phenomenon [7]. Most real-world networks show high clustering property or community structure, i.e., their nodes are organized into clusters which are also called communities [6, 8]. The clustering phenomenon indicates that the network can be divided into communities with dense connections internally and sparse connections between them. For example, people sharing the same beliefs and interests tend to connect to each other [9], and queries in the same text clustering often share similar class labels [10]. The scale-free phenomenon indicates the existence of nodes with high degrees [7], and the high degree nodes' identifiers can be also widely shared by neighbors. In the dense connected communities, as hub nodes are connected by different ones, thus the identifiers of neighbors may be used as features to capture the label patterns of nodes. Due to the widely existed high clustering property and the scale-free phenomenon in network data, we regard that the identifiers of nodes can be used as fine-grained features and community prior can be used as coarse grained prior to boost the performance of our approach in node classification tasks. In this article, we first apply IDRN [11] with 10% labeled nodes for training in the community

✉ Qi Ye
yeqi@sogou-inc.com

1 Sogou Inc., Beijing, China

detection experiments, and it improves the metric values over the existing unsupervised community detection algorithms. As well, we demonstrate the effectiveness of our algorithm on tens of public datasets which are fully labeled or partially labeled in multi-label classification tasks. In the experiments, our approach outperforms recently proposed baseline methods in most cases.

Our contributions are as follows. First, to the best of our knowledge, this is the first time that node identifiers in the egocentric networks are used as features to solve network-based classification problem. Second, we utilize the community prior in a principle way to improve its performance in different real-world networks. Finally, our approach is very effective and easy to implement, which makes it quite applicable for different real-world within-network classification tasks. The rest of the article is organized as follows. In the next section, we first review related work. Section 3 describes our methods in detail. In sect. 4, we show the experiment results in different publicly available datasets. Section 5 gives the conclusion and discussion.

2 Related Work

One of the recent focus in machine learning research is how to extend traditional classification methods to classify nodes in network data, and a body of work for this purpose has been proposed. Bhagat et al. [12] give a survey on the node classification problem in networks. They divide the methods into two categories: one uses the graph information as features and the other one propagates existing labels via random walks. The relational neighbor (RN) classifier provides a simple but effective way to solve the node classification problems. Macskassy and Provost [4] propose the weighted-vote relational neighbor (WvRN) classifier by making predictions based on the class distribution of a certain node's neighbors. It works reasonably well for within-network classification and is recommended as a baseline method for comparison. Wang and Sukthankar [5] propose a multi-label relational neighbor classification algorithm by incorporating a class propagated probability obtained from edge clustering. Macskassy and Provost [13] also believe that the very high cardinality categorical features of identifiers may cause the obvious difficulty for classifier modeling. Thus there is very little work that has incorporated node identifiers [13]. As we regard that node identifiers are also useful features for node classification, our algorithm does not solely depend on neighbors' class labels but also incorporate node identifiers in each node's egocentric networks as features and community structure as prior.

For within-network classification problem, a large number of algorithms for generating node features have been proposed. Unsupervised feature learning approaches typically exploit the spectral properties of various matrix representations of graphs. To capture different affiliations of nodes in a network, Tang and Liu [14] propose the SocioDim algorithm framework to extract latent social dimensions based on the top-d eigenvectors of the modularity matrix, and then utilize these features for discriminative learning. Rizos et al. [9] study the problem of semi-supervised, multi-label user classification of networked data in social networks. They propose a framework that combines unsupervised community extraction and supervised community-based feature weighting before training a classifier. Using the same feature learning framework, Tang and Liu [15] also propose an algorithm to learn dense features from the d-smallest eigenvectors of the normalized graph Laplacian. Ahmed et al. [16] propose an algorithm to find low-dimensional embeddings of a large graph through matrix factorization. However, the objective of the matrix factorization may not capture the global network structure information. To overcome this problem, Tang et al. [2] propose the LINE model to preserve the first-order and the second-order proximities of nodes in networks. Perozzi et al. [17] present DeepWalk which uses the SkipGram language model [18] for learning latent representations of nodes in a network by considering a set of short truncated random walks. Grover and Leskovec [19] define a flexible notion of a node's neighborhood by random walk sampling, and they propose node2vec algorithm by maximizing the likelihood of preserving network neighborhoods of nodes. Nandanwar and Murty [1] also propose a novel structural neighborhood-based classifier by random walks, while emphasizing the role of medium degree nodes in classification. As most of the algorithms based on the features generated by heuristic methods such as random walks or matrix factorization often have high time complexity, thus they may not easily be applied to large-scale real-world networks. To be more effective in node classification, in both training and prediction phrases we extract community prior and identifier features of each node in linear time, which makes our algorithm much faster. We will evaluate the scalability of our approach in different large-scale networks in the following experiments.

Several real-world network-based applications boost their performances by obtaining extra data. McDowell and Aha [20] find that accuracy of node classification may be increased by including extra attributes of neighboring nodes as features for each node. In their algorithms, the neighbors must contain extra attributes such as textual contents of web pages. Rayana and Akoglu [21] propose a framework to detect suspicious users and reviews in a user-product bipartite review network which accepts prior knowledge on the class distribution estimated from metadata. To address the problem of query classification, Bian and Chang [22] propose a label propagation method to automatically generate query class labels for unlabeled queries from click-based search logs. To identify spammer accounts, Fakhraei et al. [23] propose a statistical relational model to makes use of structural features, sequence modeling, and collective reasoning. With the help of the large amount of automatically labeled queries, the performance of the classifiers has been greatly improved. To predict the relevance issue between queries and documents, Jiang et al. [24] and Yin et al. [25] propose a vector propagation algorithm on the click graph to learn vector representations for both queries and documents in the same term space. Experiments on search logs demonstrate the effectiveness and scalability of the proposed method. Wang et al. [26] study the problem of linked document embedding for classification and propose a linked document embedding framework LDE, which combines link and label information with content information to learn document representations for classification. Newman and Clauset [27] propose a method that combines a network and its node information to detect communities. Their method learns whether the node information is correlated with the communities, and this method makes the predictions about the community membership of nodes more accurately. Tu et al. [28] propose a context-aware embedding algorithm to learn the embeddings for vertices by considering both the structural roles and text information of nodes simultaneously. Wang et al. [29] presents a novel item concept embedding approach to learn the embeddings of both items and words by leverage the concept of neighborhood proximity in both homogeneous and heterogeneous retrieval tasks. However, as it is hard to find useful extra attributes in many public available real-world network data and it may require some domain knowledge to handle the extra meta-data, in this article our approach only depends on the structural information in partially labeled networks.

3 Methodology

In this section, as a within-network classification task, we focus on performing multi-label node classification in networks, where each node can be assigned to multiple labels and only a few nodes have already been labeled. We first present our problem formulation, and then show our algorithm in detail.

3.1 Problem Formulation

The multi-label node classification we addressed here is related to the within-network classification problem: estimating labels for the unlabeled nodes in partially labeled networks. Given a partially labeled undirected network $G = \{\mathcal{V}, \mathcal{E}\}$, in which a set of nodes $\mathcal{V} = \{1, \cdots, n_{max}\}$ are connected with edge $e(i,j) \in \mathcal{E}$, and $\mathcal{L} = \{l_1, \cdots, l_{max}\}$ is the label set for nodes.

3.2 Objective Formulation

In a within-network single-label classification scenario, let Y_i be the class label variable of node i, which can be assigned to one categorical value $c \in \mathcal{L}$. Let G_i denote the information node i known about the whole graph, and let $P(Y_i = c|G_i)$ be the probability that node i is assigned to the class label c. The relational neighbor (RN) classifier is first proposed by Macskassy and Provost [4], and in the relational learning context we can get the probability $P(Y_i = c|G_i)$ by making the first-order Markov assumption [4]:

$$P(Y_i = c|G_i) = P(Y_i = c|\mathcal{N}_i), \tag{1}$$

where $\mathcal{N}_i$ is the set of nodes that are adjacent to node i. Taking advantage of the Markov assumption, Macskassy and Provost [4] proposed the weighted-vote relational neighbor (WvRN) classifier whose class membership probability can be defined as follows:

$$P(Y_i = c|G_i) = P(Y_i = c|\mathcal{N}_i) = \frac{1}{Z}\sum_{j\in\mathcal{N}_i} w_{i,j} \times P(Y_j = c|\mathcal{N}_j), \tag{2}$$

where Z is a normalizer and $w_{i,j}$ represents the weight between i and j.

3.2.1 IDRN Classifier

As shown in Eq. (2), traditional relational neighbor classifiers, such as WvRN [4], only use the class labels in neighborhood as features. However, as we will show, by taking the identifiers in each node's egocentric network as features, the classifier often performs much better than most baseline algorithms.

In our algorithm, the node identifiers, i.e., unique symbols for individual nodes, are extracted as features for learning and inference. With the first-order Markov assumption, we can simplify $G_i = G_{\mathcal{N}_i} = \mathbf{X}_{\mathcal{N}_i} = \{x|x \in \mathcal{N}_i\} \cup \{i\}$ as a feature vector of all identifiers in node i's egocentric graph $G_{\mathcal{N}_i}$. The egocentric network $G_{\mathcal{N}_i}$ of node i is the subgraph of node i's first-order zone [30]. Aside from just considering neighbors' identifiers, our approach also includes the identifier of node i itself, with the assumption that both the identifiers of node i's neighbors and itself can provide meaningful representations for its class label. For example, if node i ($ID = 1$) connects with three other nodes where $ID = 2, 3, 5$, respectively, then its feature vector $\mathbf{X}_{\mathcal{N}_i}$ of node i will be [1, 2, 3, 5]. Equation (2) can be simplified as follows:

$$P(Y_i = c|G_i) = P(Y_i = c|G_{\mathcal{N}_i}) = P(Y_i = c|\mathbf{X}_{\mathcal{N}_i}). \tag{3}$$

By taking the strong independent assumption of naive Bayes, we can simplify $P(Y_i = c|\mathbf{X}_{\mathcal{N}_i})$ in Eq. (3) as the following equation:

$$\begin{aligned} &P(Y_i = c|\mathbf{X}_{\mathcal{N}_i}) \\ &= \frac{P(Y_i = c)P(\mathbf{X}_{\mathcal{N}_i}|Y_i = c)}{P(\mathbf{X}_{\mathcal{N}_i})} \\ &\propto P(Y_i = c)P(\mathbf{X}_{\mathcal{N}_i}|Y_i = c) \\ &\propto P(Y_i = c)\prod_{k\in\mathbf{X}_{\mathcal{N}_i}} P(k|Y_i = c), \end{aligned} \tag{4}$$

where the last step drops all values independent of Y_i.

3.2.2 Multi-label Classification

Traditional ways of addressing multi-label classification problem is to transform it into a one-vs-rest learning problem [5, 14]. When training IDRN classifier, for each node i with a set of true labels T_i, we transform it into a set of single-label data points, i.e., $\{\langle \mathbf{X}_{\mathcal{N}_i}, c\rangle | c \in T_i\}$. After that, we use naive Bayes training framework to estimate the class prior $P(Y_i = c)$ and the conditional probability $P(k|Y_i = c)$ in Eq. (4).

Algorithm 1 shows how to train IDRN to get the maximal likelihood estimations (MLE) for the class prior $P(Y_i = c)$ and conditional probability $P(k|Y_i = c)$, i.e., $\hat{\theta}_c = P(Y_i = c)$ and $\hat{\theta}_{kc} = P(k|Y_i = c)$. As it has been suggested that multinomial naive Bayes classifier usually performs better than Bernoulli naive Bayes model in various real-world practices [31], we take the multinomial approach here. Suppose we observe N data points in the training dataset. Let N_c be the number of occurrences in class c and let N_{kc} be the number of occurrences of feature k and class c. In the first 2 lines, we initialize the counting values of N, N_c and N_{kc}. After that, we transform each node i with a multi-label set T_i into a set of single-label data points and use the multinomial naive Bayes framework to count the values of N, N_c and N_{kc} as shown from line 3 to line 12 in Algorithm 1. After that, we can get the estimated probabilities, i.e., $\hat{\theta}_c = P(Y_i = c)$ and $\hat{\theta}_{kc} = P(k|Y_i = c)$, for all classes and features.

In multi-label prediction phrase, the goal is to find the most probable classes for each unlabeled node. Since most methods yield a ranking of labels rather than an exact assignment, a threshold is often required. To avoid the affection of introducing a threshold, we assign s most probable classes to a node, where s is the number of labels assigned to the node originally. Unfortunately a naive implementation of Eq. (4) may fail due to numerical underflow, the value of $P(Y_i = c|\mathbf{X}_{\mathcal{N}_i})$ is proportional to the following equation:

$$P(Y_i = c|\mathbf{X}_{\mathcal{N}_i}) \propto \log P(Y_i = c) + \sum_{k\in\mathbf{X}_{\mathcal{N}_i}} \log P(k|Y_i = c). \tag{5}$$

Defining $b_c = \log P(Y_i = c) + \sum_{k\in\mathbf{X}_{\mathcal{N}_i}} \log P(k|Y_i = c)$ and using *log-sum-exp* trick [32], we get the precise probability $P(Y_i = c|\mathbf{X}_{\mathcal{N}_i})$ for each class label c as follows:

$$P(Y_i = c|\mathbf{X}_{\mathcal{N}_i}) = \frac{e^{(b_c - B)}}{\sum_{c\in\mathcal{L}} e^{(b_c - B)}}, \tag{6}$$

where $B = \max_c b_c$. Finally, to classify unlabeled nodes i, we can use Eq. (6) to assign s most probable classes to it.

Algorithm 1: Training the **Id**entifier based **r**elational **n**eighbor classifier.

```
Input: Graph $G = \{\mathcal{V}, \mathcal{E}\}$, the labeled nodes $\mathcal{V}'$ and the class label set $\mathcal{L}$.
Output: The MLE for each class $c$'s prior $\hat{\theta}_c$ and the MLE for conditional probability
        $\hat{\theta}_{kc}$.
1  $N := 0$;
2  $N_c := 0$ and $N_{kc} := 0$, $\forall c \in \mathcal{L}$ and $\forall k \in \mathcal{V}$.
3  for $i \in \mathcal{V}'$ do
4      $C = T_i$; // Get the true label set $C$ of node $i$.
5      for $c \in C$ do
6          for $k \in X_{\mathcal{N}_i}$ do
7              $N := N + 1$;
8              $N_c := N_c + 1$;
9              $N_{kc} := N_{kc} + 1$;
10         end
11     end
12 end
13 for $c \in \mathcal{L}$ do
14     $\hat{\theta}_c := \frac{N_c}{N}$;
15     for $k \in \mathcal{V}$ do
16         $\hat{\theta}_{kc} := \frac{N_{kc}+1}{N+|V|}$; // Corresponding to Laplace adding-one smoothing.
17     end
18 end
19 return $\hat{\theta}_c$ and $\hat{\theta}_{kc}$, $\forall c \in \mathcal{L}$ and $\forall k \in V$.
```

3.2.3 Community Prior

Community detection is one of the most popular topics of network science, and a large number of algorithms have been proposed recently [8, 33]. It is believed that nodes in communities share common properties or play similar roles. Grover and Leskovec [19] also regard that nodes from the same community should share similar representations. The availability of such pre-detected community structure allows us to classify nodes more precisely especially with insufficient training data. Given the community partition of a certain network, we can estimate the probability $P(Y_i = c|C_i)$ for each class c through the empirical counts and adding-one smoothing technique, where C_i indicates the community that node i belongs to. Then, we can define the probability $P(Y_i = c|\mathbf{X}_{\mathcal{N}_i})$ in Eq. (3) as follows:

$$P(Y_i = c|\mathbf{X}_{\mathcal{N}_i}, C_i) = \frac{P(Y_i = c|C_i)P(\mathbf{X}_{\mathcal{N}_i}|Y_i = c, C_i)}{P(\mathbf{X}_{\mathcal{N}_i}|C_i)}, \tag{7}$$

where $P(\mathbf{X}_{\mathcal{N}_i}|C_i)$ refers to the conditional probability of the event $\mathbf{X}_{\mathcal{N}_i}$ occurring given that node i belongs to community C_i. Obviously, given the knowledge of C_i will not influence the probability of the event $X_{\mathcal{N}_i}$ occurring, thus we can assume that $P(\mathbf{X}_{\mathcal{N}_i}|C_i) = P(\mathbf{X}_{\mathcal{N}_i})$ and $P(\mathbf{X}_{\mathcal{N}_i}|Y = c, C_i) = P(\mathbf{X}_{\mathcal{N}_i}|Y = c)$. So Eq. (7) can be simplified as follows:

$$\begin{aligned} &P(Y_i = c|\mathbf{X}_{\mathcal{N}_i}, C_i) \\ &= \frac{P(Y_i = c|C_i)P(\mathbf{X}_{\mathcal{N}_i}|Y_i = c)}{P(\mathbf{X}_{\mathcal{N}_i})} \\ &\propto P(Y_i = c|C_i)P(\mathbf{X}_{\mathcal{N}_i}|Y_i = c) \\ &\propto \log P(Y_i = c|C_i) + \sum_{k \in \mathbf{X}_{\mathcal{N}_i}} \log P(k|Y_i = c). \end{aligned} \tag{8}$$

As shown in Eq. (8), we assume that different communities have different prior rather than sharing the same global prior $P(Y_i = c)$.

3.3 Efficiency

Suppose that the largest node degree of the given network $G = \{\mathcal{V}, \mathcal{E}\}$ is K. In the training phrase, as shown in Algorithm 1, the time complexity from line 1 to line 12 is about $O(K \times |\mathcal{L}| \times |\mathcal{V}|)$, and the time complexity from line 13 to line 18 is $O(|\mathcal{L}| \times |\mathcal{V}|)$. So the total time complexity of the training phrase is $O(K \times |\mathcal{L}| \times |\mathcal{V}|)$. Obviously, it is quite simple to implement this training procedure. In the training phrase, the time complexity of each node is linear with respect to the product of the number of its degree and the size of class label set $|\mathcal{L}|$.

In the prediction phrase, suppose node i contains n neighbors. It takes $O(n+1)$ time to find its identifier vector $\mathbf{X}_{\mathcal{N}_i}$. Given the knowledge of i's community membership C_i, in Eqs. (5) and (8), it only takes $O(1)$ time to get the values of $P(Y_i = c|C_i)$ and $P(Y_i = c)$, respectively. As it takes $O(1)$ time to get the value of $P(k|Y_i = c)$, for a given class label c, the time complexities of Eqs. (5) and (8) both are $O(n)$. Thus for a given node, the total complexity of predicting the probability scores on all labels $\mathcal{L}$ is $O(|\mathcal{L}| \times n)$ even we consider predicting the precise probabilities in Eq. (6). For each class label prediction, it takes $O(n)$ time which is linear to its neighbor size. Furthermore, the prediction process can be greatly sped-up by building an inverted index of node identifiers, as the identifier features of each class label can be sparse.

4 Experiments

In this section, we first use IDRN to detect communities with partially cluster labels to show the 'community aware' characteristic it has, then we introduce the datasets and the evaluation metrics for multi-label classification tasks. After that, we conduct several experiments to show the effectiveness of our proposed algorithm. Code to reproduce our results is available at the authors' website.[1]

4.1 Networks with Known Communities

To empirically demonstrate the 'community aware' characteristic IDRN has [17], in this part, we apply IDRN on five real-world networks whose community clustering meta-data are perfectly corresponding to the underlying ground truth. Using the community meta-data as labels, we directly run IDRN without any community prior to predict the labels of unlabeled ones in these networks.

4.1.1 Metrics on Comparing Community Partitions

To quantify the similarity between the communities extracted by different algorithms and the 'ground truth' communities, we choose the normalized mutual information (NMI) [34], the Jaccard index (Jaccard) and the Rand index (Rand) as the metrics. The details of these metrics can be found in the survey [33]. All these metrics ranges from 0 when the detected community labels are uninformative to 1 when the community labels specify the original partitions completely.

[1] https://github.com/yeqi-adrs/IDRN.

4.1.2 Networks with Community Meta-data

We use the community meta-data for creating different node classes, and all these datasets are publicly available from the website.[2] The details of the datasets are summarized as follows:

- karate: The Karate club network is a well-known network of friendships between 34 members in an American University [35]. After a dispute between the coach and the treasurer, the network is further split into two communities. There are 34 nodes and 78 edges in this network.
- dolphin: The dolphin social network is an undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand [36]. The dolphin network splits into two communities as a result of the departure of a key individual. The links between nodes are established by observation of statistically significant frequent associations. The network contains 62 nodes and 159 edges.
- football: This is a collage football network which represents the schedule of games between American college football teams in the 2000 season [37]. Nodes in the network represent teams, and edges represent regular-season games between teams. There are 115 nodes and 613 edges in the network. This network is partitioned into 12 conferences.
- polbook: This is a network of books about US politics published around the time of the 2004 presidential election and sold by the online bookseller Amazon.com [38]. Edges between books represent frequent co-purchasing of books by the same buyers. The community labels are 'liberal,' 'neutral' and 'conservative.' There are 105 nodes and 441 edges in the network.
- polblog: This is a network of hyperlinks between weblogs on US politics, recorded in 2005 by Adamic and Glance [39]. Community labels are 'liberal' and 'conservative' which are assigned by blog directories or occasional self-evaluation. There are 1490 nodes and 16,715 edges in the network.

4.1.3 Experiments on Networks

To show the performance of IDRN on different networks, we randomly use 10% nodes for training and predict the community belongings of all left nodes. Please note that since we want to show the performance of IDRN on community prediction by utilizing a few community cluster labels, we just use the method in Eq. (3) without any community prior. We compare IDRN algorithm with some well-known community detection algorithms, i.e., the GN algorithm [6], the CNM algorithm [37], the Louvain algorithm [40], the MMO algorithm [41] and the label propagation algorithm (LPA) [42]. The experiments have been repeated 10 times, and then we report the average scores over different datasets in Table 1.

Table 1 shows the average NMI score, the average Jaccard index, and the average Rand index for community clustering results in the datasets. We highlight the best algorithms in each metric of different datasets. As shown in the table, IDRN outperforms most existing unsupervised baselines. As we show through our experiments in the networks with known community structure, by taking a semi-supervised approach with just a few labeled nodes, our algorithm can learn the 'community aware' characteristic better than most widely used existing unsupervised community detection algorithms.

4.2 Datasets for Classification

In the following experiments, we will predict the labels for the remaining nodes in different graph-based classification tasks. We use the following publicly available datasets described below.

Amazon	The dataset contains a subset of books from the amazon co-purchasing network data extracted by Nandanwar and Murty [1]. For each book, the dataset provides a list of other similar books, which is used to build a network. Genre of the books gives a natural categorization, and the categories are used as class labels in our experiment.
CoRA	It contains a collection of research articles in computer science domain with predefined research topic labels which are used as the ground-truth labels for each node.
IMDb	The graph contains a subset of English movies from **IMDb**,[3] and the links indicate the relevant movie pairs based on the top 5 billed stars [1]. Genre of the movies gives a natural class categorization, and the categories are used as class labels.

[2] http://www-personal.umich.edu/~mejn/netdata/.

[3] http://www.imdb.com/interfaces.

Table 1 Experiment comparisons between IDRN (with 10% labeled nodes for training) and other unsupervised community detection algorithms by the metrics of NMI, Rand index and Jaccard index

Network	Metrics (%)	Algorithms					
		IDRN	GN	CNM	Louvain	MMO	LPA
Karate	NMI	**87.62**	57.98	69.25	68.73	56.12	46.58
	Rand	**95.44**	76.92	85.90	85.90	74.36	85.90
	Jaccard	**91.58**	74.29	84.06	83.82	70.59	84.72
Dolphin	NMI	**79.88**	55.42	62.08	51.62	40.63	51.42
	Rand	**92.72**	82.39	83.23	76.73	69.18	79.25
	Jaccard	**88.40**	81.82	83.23	76.13	68.59	79.11
Football	NMI	83.67	87.89	76.24	85.61	**91.11**	82.56
	Rand	**95.47**	92.01	84.18	90.38	92.33	90.05
	Jaccard	62.25	88.84	79.49	86.89	**88.92**	86.50
Polbook	NMI	**59.87**	55.85	53.08	53.69	40.63	56.40
	Rand	**85.75**	83.67	82.77	83.22	57.60	84.35
	Jaccard	69.92	82.90	82.16	82.38	55.16	**83.61**
Polblog	NMI	**53.30**	30.34	37.99	37.55	34.77	39.09
	Rand	80.36	93.70	95.33	95.12	93.47	**95.66**
	Jaccard	67.63	93.28	95.02	94.79	93.03	**95.37**

PubMed	The dataset contains publications from **PubMed** database, and each publication is assigned to one of three diabetes classes. So it is a single-label dataset in our learning problem.
Wikipedia	The network data is a dump of Wikipedia pages from different areas of computer science. After crawling, Nandanwar and Murty [1] choose 16 top level category pages, and recursively crawled subcategories up to a depth of 3. The top level categories are used as class labels.
Youtube	A subset of Youtube users with interest grouping information is used in our experiment. The graph contains the relationships between users, and the user nodes are assigned to multiple interest groups provided by Nandanwar and Murty [1].
Blogcatalog and Flickr	These datasets are social networks, and each node is labeled by at least one category. The categories can be used as the ground truth of each node for evaluation in multi-label classification task.
PPI	It is a protein–protein interaction (PPI) network for Homo Sapiens. The labels of nodes represent the bilolgical states.
POS	This is a co-occurrence network of words appearing in the Wikipedia dump. The node labels represent Part-of-Speech (POS) tags of each word.
CiteSeer	The CiteSeer dataset consists of labeled 3312 scientific publications classified into one of six classes. The citation network consists of 4536 undirected links among the labeled nodes.
WebKB	The WebKB dataset consists of 877 scientific publications classified into one of five classes. The citation network consists of 1608 links. To form an undirected graph with all nodes labeled, there are 877 nodes and 1388 undirected edges in the graph.
SocialSpam	This anonymized dataset was collected from the Tagged.com social network website [23]. It contains 5.2 million users in the graph and 496 million undirected links between them. Each user is manually labeled as ‘spammer’ or ‘not spammer.’

$\text{Snow2014}_{\text{all}}$ and Snow2014	We use the mention and retweet social interactions to form the graph edges from the tweet collection introduced in the SNOW 2014 Data Challenge [43], and the labels belong to various types of user attribute. This dataset is a partially labeled, and we form two graphs from dataset. The $\text{Snow2014}_{\text{all}}$ graph is partially labeled with 10,992 nodes labeled, and the Snow2014 graph is a subgraph of it with all nodes labeled. Both of these two graphs are unweighted and undirected in order to make the method comparisons fair.
$\text{Youtube}_{\text{all}}$	In this graph vertices represent users in the YouTube[4] video sharing website. Apart from uploading videos, users form a subscription graph among them and also subscribe to various interest groups. There are 1,138,499 vertices and 2,990,443 edges in the original graph with only 31,703 nodes labeled.

The **Amazon**, **CoRA**, **IMDb**, **PubMed**, **Wikipedia** and **Youtube** datasets are made available by Nandanwar and Murty [1]. The **Blogcatalog** and **Flickr** datasets are provided by Tang and Liu [14], and the **PPI** and **POS** datasets are provided by Grover and Leskovec [19]. The **CiteSeer** dataset can be downloaded from the download link,[5] and the **WebKB** dataset can be fond from this link.[6] The $\text{Snow2014}_{\text{all}}$ and Snow2014 data sets can be downloaded from the link.[7] The $\text{Youtube}_{\text{all}}$ are provided by the Social Computing Data Repository at Arizona State University.[8] The **SocialSpam** dataset has been provided by the authors.[9] The statistics of the datasets are summarized in Table 2.

4.3 Classification Evaluation Metrics

In this part, we explain the details of the evaluation metrics: Hamming score, $\text{Micro}-\text{F}_1$ score and $\text{Micro}-\text{F}_1$ score which have also widely been used in many other multi-label within-network classification tasks [1, 5, 14]. Given node i, let T_i be the true label set and P_i be the predicted label set, then we have the following scores:

Definition 1 $\text{Hamming Score} = \sum_{i=1}^{|\mathcal{V}|} \frac{|T_i \cap P_i|}{|T_i \cup P_i|}$,

Definition 2

$$\text{Micro}-\text{F}_1\ \text{Score} = \frac{2\sum_{i=1}^{|\mathcal{V}|} |T_i \cap P_i|}{\sum_{i=1}^{|\mathcal{V}|} |T_i| + \sum_{i=1}^{|\mathcal{V}|} |P_i|},$$

Definition 3

$$\text{Macro}-\text{F}_1\ \text{Score} = \frac{1}{|\mathcal{L}|}\sum_{j=1}^{|\mathcal{L}|} \frac{2\sum_{i\in\mathcal{L}_j} |T_i \cap P_i|}{\sum_{i\in\mathcal{L}_j} |T_i| + \sum_{i\in\mathcal{L}_j} |P_i|},$$

where $|\mathcal{L}|$ is the number of classes and $\mathcal{L}_j$ is the set of nodes in class j.

4.3.1 Baseline Methods

In this article, we focus on comparing our work with the state-of-the-art approaches. To validate the performance of our approach, we compare our algorithms against a number of baseline algorithms. We use IDRN to denote our approach with the global priori and use IDRN_c to denote the algorithm with different community prior. All the baseline algorithms are summarized as follows:

- WvRN [4]: The **W**eighted-**v**ote **R**elational **N**eighbor is a simple but surprisingly good relational classifier. Given the neighbors $\mathcal{N}_i$ of node i, the WvRN estimates i's classification probability $P(y|i)$ of class label y with the weighted mean of its neighbors as mentioned above. As WvRN algorithm is not very complex, we implement it in Java programming language by ourselves.
- SocioDim [14]: This method is based on the SocioDim framework which generates a representation in d dimension space from the top-d eigenvectors of the modularity matrix of the network, and the eigenvectors encode the information about the community partitions of the network. The implementation of SocioDim in Matlab is available on the author's website.[10] As the authors preferred in their study, we set the number of social dimensions as 500.
- DeepWalk [17]: DeepWalk generalizes recent advancements in language modeling from sequences of words to nodes [44]. It uses local information obtained from truncated random walks to learn latent dense representations by treating random walks as the equivalent of sentences. The implementation of DeepWalk in Python has already been published by the authors.[11]

[4] https://www.youtube.com/.

[5] https://linqs-data.soe.ucsc.edu/public/lbc/citeseer.tgz.

[6] https://linqs-data.soe.ucsc.edu/public/lbc/WebKB.tgz.

[7] https://github.com/MKLab-ITI/reveal-graph-embedding.

[8] http://socialcomputing.asu.edu/datasets/YouTube2.

[9] https://linqs-data.soe.ucsc.edu/public/social_spammer/.

[10] http://leitang.net/social_dimension.html.

[11] https://github.com/phanein/deepwalk.

Table 2 Summary of undirected networks used for multi-label classification

Dataset	#Nodes	#Edges	#Classes	Average category	$\frac{\#\text{Edges}}{\#\text{Nodes}}$
Amazon	83,742	190,097	30	1.546	2.270
CoRA	24,519	92,207	10	1.004	3.782
IMDb	19,359	362,079	21	2.301	18.703
PubMed	19,717	44,324	3	1.000	2.248
Wikipedia	35,633	495,388	16	1.312	13.903
Youtube	22,693	96,361	47	1.707	4.246
Blogcatalog	10,312	333,983	39	1.404	32.387
Flickr	80,513	5,899,882	195	1.338	73.278
PPI	3890	37,845	50	1.707	9.804
POS	4777	92,295	40	1.417	19.320
CiteSeer	3312	4536	6	1.000	1.369
Snow2014	9489	22,309	90	2.351	2.538
WebKB	877	1388	5	1.000	1.582
SocialSpam	5,275,125	49,6691,571	2	1.000	94.157
Snow2014$_{\text{all}}$	533,874	942,226	90	2.534	1.764
Youtube$_{\text{all}}$	1,138,499	2,990,443	47	1.000	2.626

- LINE [2]: LINE algorithm proposes an approach to embed networks into low-dimensional vector spaces by preserving both the *first-order* and *second-order* proximities in networks. The implementation of LINE in C++ has already been published by the authors.[12] To enhance the performance of this algorithm, we set embedding dimensions as 256 (i.e., 128 dimensions for the *first-order* proximities and 128 dimensions for the *second-order* proximities) in LINE algorithm as preferred in its implementation.
- SNBC [1]: To classify a node, SNBC takes a structured random walk from the given node and makes a decision based on how nodes in the respective k^{th}-level neighborhood are labeled. The implementation of SNBC in Matlab has already been published by the authors.[13]
- node2vec [19]: It also takes a similar approach with DeepWalk which generalizes recent advancements in language modeling from sequences of words to nodes. With a flexible neighborhood sampling strategy, node2vec learns a mapping of nodes to a low-dimensional feature space that maximizes the likelihood of preserving network neighborhoods of nodes. The implementation of node2vec in Python is available on the authors' website.[14]

We obtain 128 dimension embeddings for a node using DeepWalk and node2Vec as preferred in the algorithms. After getting the embedding vectors for each node, we use these embeddings further in classification. In the multi-label classification experiment, each node is assigned to one or more class labels. We assign s most probable classes to the node using these decision values, where s is equal to the number of labels assigned to the node originally. Specifically, for all vector representation models (i.e., SocioDim, DeepWalk, LINE, SNBC and node2vec), we use a one-vs-rest logistic regression implemented by LibLinear [45] to return the most probable labels as described in prior work [5, 14, 17]. In the following parts, we will evaluate the performances of within-network classifiers in different datasets, respectively. As some baseline algorithms are just designed for undirected or unweighted graphs, we transform all the graphs to undirected and unweighted ones for a fair comparison.

4.4 Different Community Prior

To show the performance of IDRN with different community prior from various community detection algorithms, we combine IDRN with different community detection algorithms, i.e., the CNM algorithm [37], the Louvain algorithm [40], the MMO algorithm [41] and the label propagation algorithm (LPA) [42]. Table 3 shows the metrics of IDRN with 10% nodes labeled with underlying cluster labels for training and the rest for testing. As shown in the table, IDRN with different community prior improve the metrics over the one with global prior. The results also indicate that there is no much difference between the prior got by different community detection algorithms. Although IDRN with the CNM algorithm seems to be comparable with IDRN with the Louvain algorithm, however, to keep

[12] https://github.com/tangjianpku/LINE.

[13] https://github.com/sharadnandanwar/snbc.

[14] https://github.com/aditya-grover/node2vec.

Table 3 Experiment comparisons of IDRN with different community prior by the metrics of Hamming score, Micro $-$ F_1 score and Macro $-$ F_1 score with 10% nodes labeled for training

Metric	Network	IDRN	$\text{IDRN}_{\text{Louvain}}$	IDRN_{CNM}	IDRN_{MMO}	IDRN_{LPA}
Hamming score (%)	Amazon	56.20	63.86	63.54	56.09	58.65
	Youtube	37.17	39.91	39.94	38.90	39.58
	CoRA	71.71	73.58	72.33	71.77	72.33
	IMDb	20.33	20.79	21.20	20.97	21.00
	PubMed	73.76	78.02	78.07	74.57	76.14
	Wikipedia	71.67	71.12	70.54	72.02	72.02
	Flickr	28.44	29.11	29.62	29.66	29.71
	Blogcatalog	26.77	26.29	26.39	26.13	26.73
	PPI	11.39	12.34	11.77	12.06	12.22
	POS	39.09	39.94	39.62	39.14	39.12
	CiteSeer	43.55	55.76	54.42	46.22	49.21
	Snow2014	24.97	27.50	26.80	25.37	25.95
	WebKB	47.17	44.25	46.01	48.37	44.01
Micro-F_1 (%)	Amazon	56.86	64.62	64.32	57.07	59.64
	Youtube	43.08	45.20	45.20	44.27	45.03
	CoRA	71.72	73.59	72.34	71.78	72.34
	IMDb	29.63	29.81	30.29	29.94	29.98
	PubMed	73.76	78.02	78.07	74.57	76.14
	Wikipedia	73.21	72.79	72.18	73.57	73.51
	Flickr	31.99	32.56	33.09	33.03	33.09
	Blogcatalog	29.05	28.81	28.80	28.39	29.06
	PPI	15.82	17.15	16.21	16.64	16.74
	POS	42.36	43.71	43.29	42.48	42.56
	CiteSeer	43.55	55.76	54.42	46.22	49.21
	Snow2014	28.87	30.55	30.24	29.09	28.94
	WebKB	47.17	44.25	46.01	48.37	44.01
Macro-F_1 (%)	Amazon	53.48	61.00	61.03	53.91	57.57
	Youtube	34.48	37.85	36.84	35.06	37.28
	CoRA	64.57	66.36	64.60	64.54	65.80
	IMDb	19.96	20.57	20.57	20.14	20.43
	PubMed	72.04	76.79	76.89	72.86	74.64
	Wikipedia	64.84	65.46	64.40	65.29	64.64
	Flickr	14.85	14.80	15.17	14.91	15.68
	Blogcatalog	11.39	12.08	11.74	11.32	11.56
	PPI	10.93	12.96	12.33	12.53	11.88
	POS	5.88	6.68	6.65	5.87	6.12
	CiteSeer	40.36	52.05	50.43	43.08	46.32
	Snow2014	14.00	17.82	17.51	14.42	17.02
	WebKB	26.38	26.01	26.76	25.44	25.74

the balance of the speed and performance, we still choose the Louvain algorithm to get the community prior of IDRN in the following experiments as suggested by the experiments in different papers [8, 33].

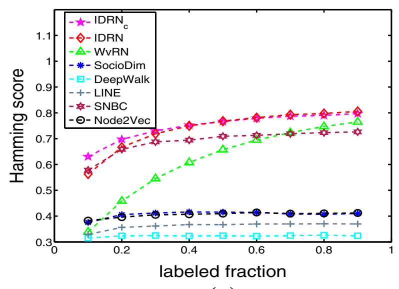
4.5 Performance of Classifiers

To study the performance of different classification algorithms on a sparsely labeled network, we show results obtained by using 10% nodes for training and the left 90% nodes for testing. The process has been repeated 10 times, and we report the average scores over different datasets. As mentioned above, we choose the Louvain algorithm to extract communities in networks which has been shown as one of the best performing algorithms. This community detection algorithm has also been implemented by us and is available at the authors' website.

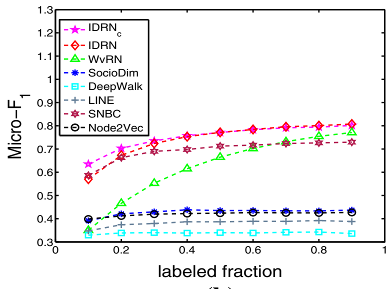
Table 4 Experiment comparisons of baselines, IDRN and IDRN$_c$ by the metrics of Hamming score, Micro $-$ F$_1$ score and Macro $-$ F$_1$ score with 10% labeled nodes for training

Metric	Network	WvRN	SocioDim	DeepWalk	LINE	SNBC	node2vec	IDRN	IDRN$_c$
Hamming score (%)	Amazon	33.76	38.36	31.79	40.55	59.00	49.18	56.20	**63.86**
	Youtube	22.82	31.94	36.63	33.90	35.06	33.86	37.17	**39.91**
	CoRA	55.83	63.02	71.37	65.50	66.75	72.66	71.71	**73.58**
	IMDb	**33.59**	22.21	33.12	30.39	30.18	32.97	20.33	20.79
	PubMed	50.32	65.68	77.40	68.31	**79.22**	79.02	73.76	78.02
	Wikipedia	45.10	65.29	71.10	68.812	68.78	70.69	**71.67**	71.121
	Flickr	21.37	29.67	28.73	30.96	24.20	30.65	28.44	29.11
	Blogcatalog	17.89	27.04	25.63	25.32	22.40	**27.46**	26.77	26.29
	PPI	6.28	8.61	8.14	9.27	7.97	8.88	11.39	**12.34**
	POS	23.05	21.06	31.40	38.24	37.73	34.59	39.09	**39.94**
	CiteSeer	31.89	20.04	23.80	22.79	54.89	50.99	43.44	**55.76**
	Snow2014	15.71	9.39	8.71	12.97	22.57	20.00	24.91	**27.50**
	WebKB	42.67	28.06	27.58	45.57	47.97	35.79	**48.34**	44.25
	SocialSpam	11.29	–	–	–	–	–	98.89	**98.91**
Micro-F_1 (%)	Amazon	34.86	39.62	33.06	42.42	59.79	50.55	56.86	**64.62**
	Youtube	27.81	36.40	40.73	38.01	39.67	38.35	43.08	**45.20**
	CoRA	55.85	63.00	71.36	65.47	66.78	72.66	71.72	**73.59**
	IMDb	**42.62**	29.99	41.82	39.89	39.53	42.36	29.63	29.81
	PubMed	50.32	65.68	77.40	68.31	**79.22**	79.02	73.76	78.02
	Wikipedia	48.51	66.95	72.19	70.21	70.68	72.07	**73.21**	72.79
	Flickr	25.40	32.91	31.66	**34.03**	27.60	33.76	31.99	32.56
	Blogcatalog	20.50	28.86	27.29	27.45	24.66	**29.41**	29.05	28.81
	PPI	**18.41**	12.29	11.52	13.16	11.32	12.80	15.82	17.15
	POS	26.04	24.42	35.98	42.70	41.99	39.09	42.36	**43.71**
	CiteSeer	31.89	20.04	23.80	22.79	54.89	50.99	43.44	**55.76**
	Snow2014	20.43	15.23	14.50	17.26	23.36	24.55	29.02	**30.55**
	WebKB	42.67	28.06	27.58	45.57	47.97	35.79	**48.34**	44.25
	SocialSpam	11.29	–	–	–	–	–	98.89	**98.91**
Macro-F_1 (%)	Amazon	32.00	35.95	21.64	37.52	56.84	45.85	53.48	**61.00**
	Youtube	18.17	34.19	33.92	33.47	32.07	32.60	34.48	**37.85**
	CoRA	43.16	56.82	62.68	59.07	55.68	64.79	64.57	**66.36**
	IMDb	18.89	18.77	18.22	18.83	17.45	18.46	19.96	**20.57**
	PubMed	41.57	64.85	75.92	66.66	77.16	**77.50**	72.04	76.79
	Wikipedia	45.58	58.93	62.29	62.17	61.99	64.90	64.84	**65.46**
	Flickr	15.54	18.28	17.13	**21.80**	7.36	18.46	14.85	14.80
	Blogcatalog	11.47	**18.88**	14.65	15.52	8.29	17.16	11.39	12.08
	PPI	7.35	10.59	9.61	10.82	8.27	11.27	10.93	**12.96**
	POS	3.91	6.05	8.26	**8.93**	5.92	8.61	5.58	6.68
	CiteSeer	26.13	18.27	21.84	20.26	51.17	46.39	40.50	**52.05**
	Snow2014	9.67	4.41	4.53	8.84	12.57	14.03	14.06	**17.82**
	WebKB	15.86	20.19	20.54	24.53	25.53	20.05	**26.31**	26.01
	SocialSpam	10.68	–	–	–	–	–	98.75	**98.77**

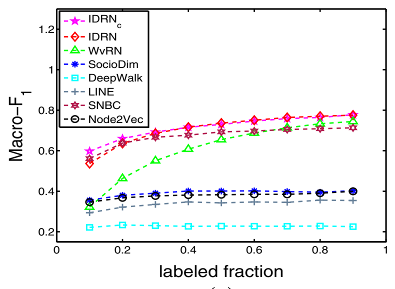
Table 4 shows the average metric scores for multi-label classification results in the datasets. We highlight the best performance algorithms of each metric in bold. As shown in the table, in most of the cases, IDRN and IDRN$_c$ algorithms improve the metrics over the existing baselines. Our model with community prior, i.e., IDRN$_c$ often performs



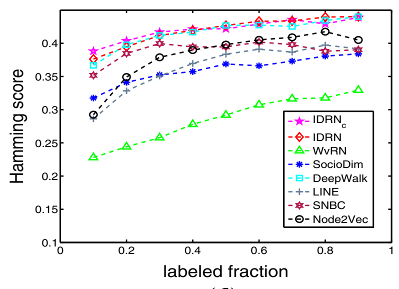
(a)



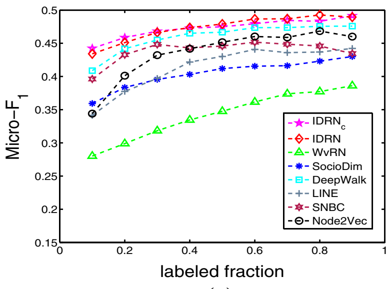
(b)



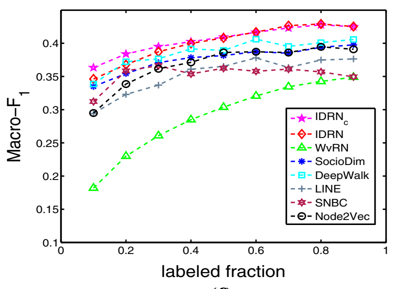
(c)



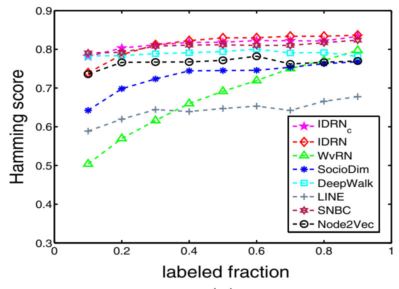
(d)



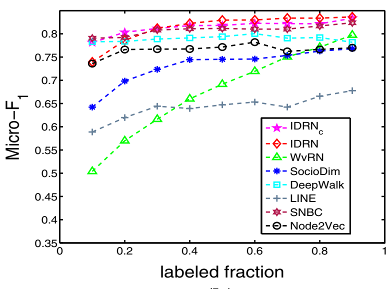
(e)



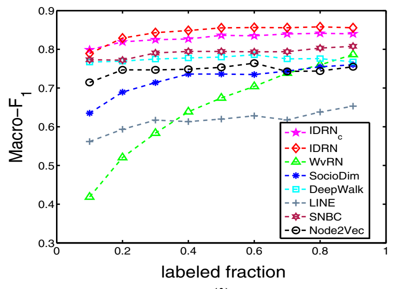
(f)



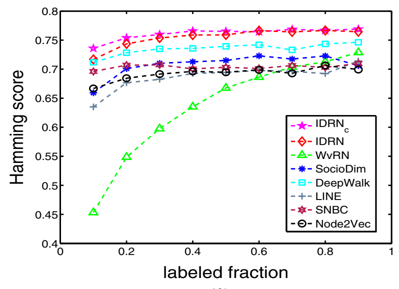
(g)



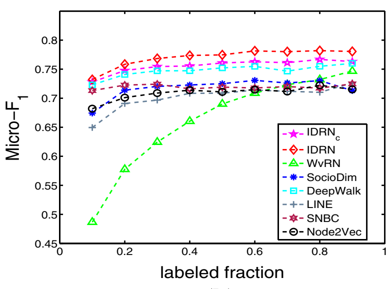
(h)



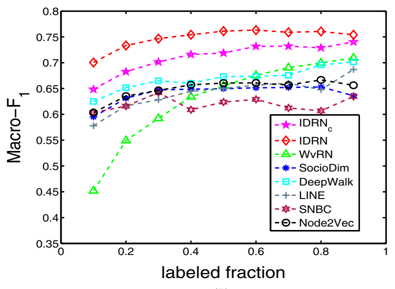
(i)



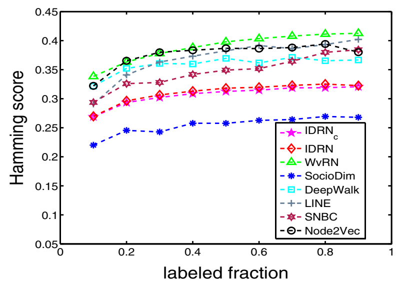
(j)



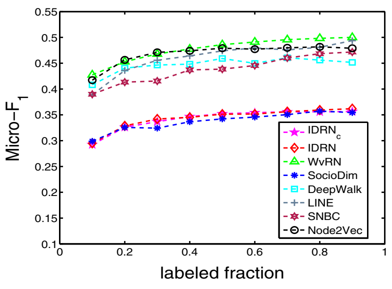
(k)



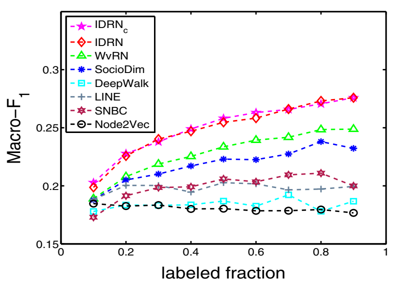
(l)



(m)



(n)



(o)

◀**Fig. 1** Performance evaluation of Hamming scores, Micro $-$ F_1 scores and Macro $-$ F_1 scores on varying the amount of labeled data used for training. The x axis denotes the fraction of labeled data, and the y axis denotes the Hamming scores, Micro $-$ F_1 scores and Macro $-$ F_1 scores, respectively. **a** Amazon. **b** Amazon. **c** Amazon. **d** Youtube. **e** Youtube. **f** Youtube. **g** PubMed. **h** PubMed. **i** PubMed. **j** Wikipedia. **k** Wikipedia. **l** Wikipedia. **m** IMDB. **n** IMDB. **o** IMDB

better than IDRN with global prior. For the three metrics, $IDRN_c$ performs consistently better than other algorithms in most of these datasets. Take **IMDb** dataset for an example, we observe that Hamming score and Micro $-$ F_1 score got by $IDRN_c$ are worse than those got by some baseline algorithms, such as node2vec and WvRN, however Macro $-$ F_1 score got by $IDRN_c$ is the best. As Macro $-$ F_1 score computes an average over classes while Hamming and Micro $-$ F_1 scores get the average over all testing nodes, the result may indicate that our algorithms get more accurate results over different classes in the imbalanced **IMDb** dataset. To show the results more clearly, we also get the average validation scores for each algorithm in these datasets which are shown in the last lines of the three metrics in Table 4. On average our approach can provide Hamming score, Micro $-$ F_1 score and Macro $-$ F_1 score higher than competing methods. The results indicate that our $IDRN_c$ outperforms most baseline methods when networks are sparsely labeled. We also perform our algorithm on an extremely large dataset—**SocialSpam**. As **SocialSpam** is very huge, most of the competing methods cannot finish their algorithm in less than 24 hours which are indicated by the '-' characters in the table. Among all the competing methods only the simplest algorithm, i.e., WvRN can classify the nodes in **SocialSpam** in 24 hours. However, our results are much better than those got by WvRN, and the Hamming score, Micro $-$ F_1 score and Macro $-$ F_1 score got by $\mathbf{IDRN_c}$ are 98.91, 98.91 and 98.77%, respectively. We believe the reason is that since the **SocialSpam** network is much denser than others and it can provide more node identifiers as features, so our algorithms works very well in such dense and large networks.

Second, we show the performance of the classification algorithms of different training fractions. When training a classifier, we randomly sample a portion of the labeled nodes as the training data and the rest as the test. For all the datasets, we randomly sample 10–90% of the nodes as the training samples, and use the left nodes for testing. The process has been repeated 5 times, and we report the averaged scores. Due to limitation in space, we just summarize the results of some datasets for Hamming scores, Micro $-$ F_1 scores and Macro $-$ F_1 scores in Fig. 1. Here we can make similar observations with the conclusion given in Table 4. As shown in Fig. 1, IDRN and $IDRN_c$ perform consistently better than other algorithms in the **Amazon**, **Youtube**, **PubMed** and **Wikipedia** datasets in Fig. 1. In fact, nearly in all these datasets, our approaches outperform all the baseline methods significantly, as the lack of space we do not show all the figures here. Note in the **IMDb** dataset as shown in Fig. 1m, n the Hamming scores and Micro $-$ F_1 scores got by IDRN and $IDRN_c$ are not the best, however, the Macro $-$ F_1 scores got by IDRN and $IDRN_c$ are much better than those got by other classification algorithms as shown in Fig. 1o. The results also indicate that our algorithms get more accurate results over different classes in the imbalanced dataset. We also note that when the networks are sparsely labeled (i.e., with 10 or 20% labeled data), $IDRN_c$ performs slightly better than IDRN. However, when more nodes are labeled, IDRN usually outperforms $IDRN_c$. As we see that the posterior in Eq. (3) is a combination of prior and likelihood, the results may indicate that the community prior of a given node corresponds to a strong prior, while the global prior is a weak one. The strong prior will improve the performance of IDRN when the training fractions are small, while the opposite conclusion holds for training on large training fractions.

As an empirical benchmark, we perform a scalability test to evaluate the running time of our algorithms in large-scale networks in Table 5. We show the results in the networks with most edges using 10% training data and the rest for testing data. Both IDRN and $IDRN_c$ are written in Java, and these algorithms are performed on a server with 256 G memory, Intel Xeon 2.60 GHz CPU, and Redhat OS system in a single thread. As shown in Table 5, IDRN and $IDRN_c$ are very fast to handle large-scale networks which is in accordance with the complexity analysis we mention above. Both of our methods can finish their training and testing processes in nearly one hour in the largest network, **SocialSpam**, which contains 5.2 million users and 496 million links. However, it may be even hard for most other comparing algorithms to load such large-scale networks in memory.

4.6 Performance of Classifiers in Partially Labeled Networks

As we have mentioned before, in many real-world applications, the lack of labeled data poses a major practical

Table 5 Summary of running time (in s) of IDRN and $IDRN_c$ in each validation process by using 10% training data and 90% testing data

Dataset	Wikipedia	$Snow2014_{all}$	$Youtube_{all}$	Flickr	Spammer
IDRN	2	3	14	518	2937
$IDRN_c$	2	5	9	482	4145

Table 6 Experiment comparisons of baselines, IDRN and IDRN$_c$ by the metrics of Hamming score, Micro $-$ F$_1$ score and Macro $-$ F$_1$ score with 10% labeled nodes for training in partially labeled graphs

Metric	Network	WvRN	DeepWalk	LINE	node2vec	IDRN	IDRN$_c$
Hamming score (%)	Youtube$_{all}$	24.05	10.32	27.51	33.21	**39.16**	35.02
	Snow2014$_{all}$	15.78	7.24	21.76	20.64	22.49	**24.76**
Micro-F_1 (%)	Youtube$_{all}$	37.42	15.18	32.95	38.38	**60.46**	41.05
	Snow2014$_{all}$	20.53	12.06	25.62	26.45	25.63	**27.23**
Macro-F_1 (%)	Youtube$_{all}$	30.94	8.79	25.90	32.34	**60.38**	33.95
	Snow2014$_{all}$	8.67	4.37	15.97	16.05	14.73	**18.69**

issue for many network classification tasks. In most cases, large-scale networks may only be partially labeled in real-world tasks, so it is valuable to show the performance of our algorithms in these partially labeled datasets. In this part, we evaluate our algorithms in two partially labeled networks, i.e, Youtube$_{all}$ and Snow2014$_{all}$, which are not used in above experiments. We run some of the comparing methods which are capable to handle these datasets as baselines. More precisely, after getting the structural features by the algorithms, we use 10% labeled data for training and use the left 90% for testing just as the method used by others [9]. As shown in Table 6, in these partially labeled networks, the result still suggests that our algorithms work better than most comparing methods. For example, in the Youtube$_{all}$ network, IDRN outperforms all baselines by at least 17.91, 57.52 and 86.70% with respect to Hamming score, Macro $-$ F$_1$ score and Micro $-$ F$_1$ score, respectively.

5 Conclusion and Discussion

In this article, we propose a novel approach for node classification, which uses the node identifiers in the egocentric networks as fine-grained likelihood features and community prior. Using the coarse-grained community prior, we can get high level features of the nodes' categories. However, these coarse-grained features may not be discriminative enough especially for the nodes linked to different communities. As node identifiers can be shared by neighbors, identifiers retain fine-grained details critical for discrimination class labels in nodes' adjacent communities. Thus, we propose our new approach which combines the coarse-grained features and fine-grained features. We consider that this is the first time that node identifiers in the egocentric networks are used as features. In this article, first, we show that IDRN can learn the 'community aware' characteristic, and it stably outperforms most existing unsupervised community detection algorithms with a few underlying cluster labels. After that, empirical evaluation confirms that our proposed algorithm is capable of handling high-dimensional identifier features and achieves better performance in real-world networks. We demonstrate the effectiveness of our approach on many publicly available datasets. No matter networks are sparsely labeled or densely labeled, our approach usually provides higher metric scores than competing methods. Moreover, our method is quite practical and efficient, since it only requires the features extracted from network structure without any extra data which makes it suitable for different real-world within-network classification tasks.

It should be noted that there is significant space for our future research. We would like to assess our algorithm's effectiveness in classifying search queries in click-through bipartite graph by integrating extra textual features in web search companies. Furthermore, as many real-world network, e.g., social networks and click-through networks, are evolving over time, it is important to classify these newly appearing ones automatically. We could improve our algorithm to predict the node labels in these involving networks in the future. Besides, as we find that the node identifiers are very useful features, we would like to try more effective classification models which can combine these sparse, high-dimensional and discrete features, such as node identifiers, with other deep representations in a low-dimensional space to get higher classification performance.

Acknowledgements The authors would like to thank all the members in ADRS (ADvertisement Research for Sponsered search) group in Sogou Inc. for the help with parts of the data processing and experiments.

References

1. Nandanwar S, Murty MN (2016) Structural neighborhood based classification of nodes in a network. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, San Francisco, CA, USA, August 13–17, 2016, ACM, pp 1085–1094

2. Tang J, Qu M et al (2015) LINE: large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, WWW'15, pp 1067–1077
3. Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: Proceedings of the 22Nd ACM SIGKDD international conference on knowledge discovery and data mining (ACM, New York, NY, USA), KDD'16, pp 1225–1234
4. Macskassy SA, Provost F (2003) A simple relational classifier. In: Proceedings of the second workshop on multi-relational data mining (MRDM-2003) at KDD-2003, pp 64–76
5. Wang X, Sukthankar G (2013) Multi-label relational neighbor classification using social context features. In: Proceedings of the 19th ACM SIGKDD conference on knowledge discovery and data mining (KDD) (Chicago, USA), pp 464–472
6. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99(12):7821–7826
7. Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286:509–512
8. Fortunato S, Hric D (2016) Community detection in networks: a user guide. Phys Rep 659:1–44
9. Rizos G, Papadopoulos S, Kompatsiaris Y (2017) Collective spammer detection in evolving multi-relational social networks. PLoS ONE 12(3):e0173347
10. Ye Q, Wang F, Bo L (2016) StarrySky: a practical system to track millions of high-precision query intents. In: Proceedings of the 25th international conference companion on World Wide Web (WWW'16 companion). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, pp 961–966. https://doi.org/10.1145/2872518.2890588
11. Ye Q, Zhu C, Li G, Wang F (2017) Combining node identifier features and community priors for within-network classification. In: Asia-Pacific Web (APWeb) and web-age information management (WAIM) joint conference on web and big data part II, Springer, pp 3–17
12. Bhagat S, Cormode G, Muthukrishnan S (2011) Node classification in social networks. CoRR arXiv.1101.3291
13. Macskassy SA, Provost F (2007) Classification in networked data: a toolkit and a univariate case study. J Mach Learn Res 8(May):935–983
14. Tang L, Liu H (2009) Relational learning via latent social dimensions. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining (ACM, New York, NY, USA), KDD'09, pp 817–826
15. Tang L, Liu H (2009) Scalable learning of collective behavior based on sparse social dimensions. In: The 18th ACM conference on information and knowledge management ACM. NY, USA, New York, pp 1107–1116
16. Ahmed A, Shervashidze N, et al. (2013) Distributed large-scale natural graph factorization, In: Proceedings of the 22nd international conference on World Wide Web, ACM, pp 37–48
17. Perozzi B, Al-Rfou R, Skiena S (2014) DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining (ACM, New York, NY, USA), KDD'14, pp 701–710
18. Joulin A, Grave E et al (2016) Bag of tricks for efficient text classification. CoRR arXiv:1607.01759
19. Grover A, Leskovec J (2016) Node2Vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (ACM, New York, NY, USA), KDD'16, pp 855–864
20. McDowell LK, Aha DW (2013) Labels or attributes? Rethinking the neighbors for collective classification in sparsely-labeled networks, In: International conference on information and knowledge management. ACM Press (ACM Press, San Francisco, CA), pp 847–852
21. Rayana S, Akoglu L (2015) Collective opinion spam detection: bridging review networks and metadata. In: Proceedings of the 21st ACM SIGKDD international conference on knowledge discovery and data mining (ACM), pp 985–994
22. Bian J, Chang Y (2011) A taxonomy of local search: semi-supervised query classification driven by information needs. In: Proceedings of the 20th ACM international conference on information and knowledge management (ACM, New York, NY, USA), CIKM'11, pp 2425–2428
23. Fakhraei S, Foulds J et al (2015) Multilabel user classification using the community structure of online networks. PLoS ONE, KDD'15, pp 1769C1778. ACM
24. Jiang S, Hu Y etal (2016) Learning query and document relevance from a web-scale click graph. In: Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval (ACM, New York, NY, USA), SIGIR'16, pp 185–194
25. Yin D, Hu Y et al (2016) Ranking relevance in yahoo search. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (ACM, New York, NY, USA), KDD'16, pp 323–332
26. Wang S, Tang J et al (2016) Linked document embedding for classification. In: Proceedings of the 25th ACM international on conference on information and knowledge management (ACM, New York, NY, USA), CIKM'16, pp 115–124
27. Newman ME, Clauset A (2016) Structure and inference in annotated networks. Nat Commun 7:11863
28. Tu C, Liu H, Liu Z, Sun M (2017) CANE: context-aware network embedding for relation modeling. In: Proceedings of the 55th annual meeting of the association for computational linguistics, pp 1722–1731
29. Wang CJ, Wang TH et al (2017) ICE: item concept embedding via textual information. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval (ACM), pp 85–94
30. Marsden PV (2002) Egocentric and sociocentric measures of network centrality. Soc Netw 24(4):407–422
31. Wang SI, Manning CD (2012) Baselines and bigrams: simple, good sentiment and topic classification. In: Proceedings of the ACL, pp 90–94
32. Murphy KP (2012) Machine learning: a probabilistic perspective. The MIT Press, Cambridge
33. Fortunato S (2010) Community detection in graphs. Phys Rep 486(3–5):75–174
34. Danon L, Duch J, Arenas A, Daz-guilera A (2005) Comparing community structure identification. J Stat Mech Theory Exp 9008:09008
35. Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33(4):452–473
36. Lusseau D, Schneider K et al (2003) The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. Behav Ecol Sociobiol 54(4):396
37. Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70(6):066111
38. Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world' networks. Nature 393:440–442
39. Adamic LA, Glance N (2005) The political blogosphere and the 2004 US election: divided they blog. In: Proceedings of the 3rd international workshop on Link discovery (ACM), pp 36–43
40. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):P10008
41. Ye Q, Bin W, Bai W (2013) The influence of technology on social network analysis and mining (Springer), vol 6, chap. 16 detecting communities in massive networks efficiently with flexible resolution, pp 373–392

42. Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76(3):036106
43. Papadopoulos S, Corney D, Aiello LM (2014) SNOW 2014 data challenge: assessing the performance of news topic detection methods in social media. In: SNOW-DC@ WWW, pp 1–8
44. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. CoRR arXiv:1301.3781
45. Fan RE, Chang KW et al (2008) LIBLINEAR: a library for large linear classification. J Mach Learn Res 9:1871–1874