Combining Node Identifier Features and Community Priors for Within-Network Classification

Qi Ye(✉), Changlei Zhu, Gang Li, and Feng Wang

Sogou Inc., Beijing, China
{yeqi,zhuchanglei,ligang,wangfeng}@sogou-inc.com

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 paper, 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. On average, our approach can provide Hamming score, Micro-F_1 score and Macro-F_1 score up to 14%, 21% and 14% higher than competing methods respectively in sparsely labeled 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 their large size, and 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 unlabelled nodes. Recently, much attention has been paid to this problem, and various prediction algorithms over nodes have been proposed [19,22,25].

In this paper, we propose a within-network classifier which makes use of the first-order Markov assumption that labels of each node are only dependent on its neighbors and itself. Traditional relational classification algorithms, such as WvRn [13] and SCRN [27] 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 [9] and scale-free phenomenon [2]. Most real-world networks show high clustering property or community structure, i.e., their nodes are organized into clusters which are also called communities [8,9]. The clustering phenomenon indicates that the network can be divided into communities with dense connections internally and sparse connections between them. In the dense connected communities, the identifiers of neighbors may capture link patterns between nodes. The scale-free phenomenon indicates the existence of nodes with high degrees [2], and we regard that the identifiers of these high degree nodes can also be useful to capture local patterns. By introducing the node identifiers as fine-grained features, we propose **id**entifier based **r**elational **n**eighbor classifier (IDRN) by incorporating the first Markov assumption and community priors. As well, we demonstrate the effectiveness of our algorithm on 10 public datasets. In the experiments, our approach outperforms some recently proposed baseline methods.

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 priors to improve its performance in sparsely labeled networks. Finally, our approach is very effective and easily to implement, which makes it quite applicable for different real-world within-network classification tasks. The rest of the paper 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. [3] 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 propagate 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 [13] 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 [27] propose a multi-label relational neighbor classification algorithm by incorporating a class propagated probability obtained from edge clustering. Macskassy et al. [14] 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 [14]. 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 incorporating local node identifiers as features and community structure as priors.

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 [23] 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. Using the same feature learning framework, Tang and Liu [24] also propose an algorithm to learn dense features from the d-smallest eigenvectors of the normalized graph Laplacian. Ahmed et al. [1] 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. [22] propose the LINE model to preserve the first-order and the second-order proximities of nodes in networks. Perozzi et al. [20] present DeepWalk which uses the SkipGram language model [12] for learning latent representations of nodes in a network by considering a set of short truncated random walks. Grover and Leskovec [10] 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 [19] also propose a novel structural neighborhood-based classifier by random walks, while emphasizing the role of medium degree nodes in classification. As 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.

Several real-world network based applications boost their performances by obtaining extra data. McDowell and Aha [16] 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 contains 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 [4] propose a label propagation method to automatically generate query class labels for unlabeled queries from click-based search logs. 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. [11] and Yin et al. [28] 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. As it is hard to find useful extra attributes in many real-world networks, 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 details.

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 [13], and in the relational learning context we can get the probability $P(Y_i = c|G_i)$ by making the first order Markov assumption [13]:

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

where $\mathcal{N}_i$ is the set of nodes that are adjacent to node i. Taking advantage of the Markov assumption, Macskassy and Provost [13] 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.

IDRN Classifier. As shown in Eq. 2, traditional relational neighbor classifiers, such as WvRN [13], 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 [15]. 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 $[1, 2, 3, 5]$. Eq. 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.

Multi-label Classification. Traditional ways of addressing multi-label classification problem is to transform it into a one-vs-rest learning problem [23,27]. 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 [26], 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 [18], 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 the 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 $k \in \boldsymbol{X}_{\mathcal{N}_i}$ 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 $\forall k \in V$.
```

Community Priors. Community detection is one of the most popular topics of network science, and a large number of algorithms have been proposed recently [7,8]. It is believed that nodes in communities share common properties or play similar roles. Grover and Leskovec [10] 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 priors rather than sharing the same global prior $P(Y_i = c)$. To extract communities in networks, we choose the Louvain algorithm [5] in this paper which has been shown as one of the best performing algorithms.

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 introduce the dataset and the evaluation metrics. After that, we conduct several experiments to show the effectiveness of our algorithm. Code to reproduce our results will be available at the authors' website[1].

4.1 Dataset

The task is to predict the labels for the remaining nodes. 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 [19]. 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**[2], and the links indicate the relevant movie pairs based on the top 5 billed stars [19]. Genre of the movies gives a natural class categorization, and the categories are used as class labels.

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 [19] 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.

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.

[1] https://github.com/yeqi-adrs/IDRN.
[2] http://www.imdb.com/interfaces.

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.

The **Amazon**, **CoRA**, **IMDb**, **PubMed**, **Wikipedia** and **Youtube** datasets are made available by Nandanwar and Murty [19]. The **Blogcatalog** and **Flickr** datasets are provided by Tang and Liu [23], and the **PPI** and **POS** datasets are provided by Grover and Leskovec [10]. The statistics of the datasets are summarized in Table 1.

Table 1. Summary of undirected networks used for multi-label classification.

Dataset	#Nodes	#Edges	#Classes	Average Category	$\frac{\#\textbf{Edges}}{\#\textbf{Nodes}}$
Amazon	83742	190097	30	1.546	2.270
CoRA	24519	92207	10	1.004	3.782
IMDb	19359	362079	21	2.301	18.703
PubMed	19717	44324	3	1.000	2.248
Wikipedia	35633	495388	16	1.312	13.903
Youtube	22693	96361	47	1.707	4.246
Blogcatalog	10312	333983	39	1.404	32.387
Flickr	80513	5899882	195	1.338	73.278
PPI	3890	37845	50	1.707	9.804
POS	4777	92295	40	1.417	19.320

4.2 Evaluation Metrics

In this part, we explain the details of the evaluation metrics: Hamming score, Micro-F$_1$ score and Macro-F$_1$ score which have also widely been used in many other multi-label within-network classification tasks [19,23,27]. 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. *Hamming Score* $= \sum_{i=1}^{|\mathcal{V}|} \frac{|T_i \cap P_i|}{|T_i \cup P_i|}$,

Definition 2. *Micro-F$_1$ 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. *Macro-F$_1$ 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.

Baseline Methods. In this paper, 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. In this paper, we use IDRN to denote our approach with the global priori and use IDRN$_c$ to denote the algorithm with different community priors. All the baseline algorithms are summarized as follows:

- WvRN [13]: 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 [23]: 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 web-site[3]. As the authors preferred in their study, we set the number of social dimensions as 500.
- DeepWalk [20]: DeepWalk generalizes recent advancements in language modeling from sequences of words to nodes [17]. 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[4].
- LINE [22]: 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[5]. 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 [19]: 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[6].
- node2vec [10]: 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' web-site[7].

[3] http://leitang.net/social_dimension.html.
[4] https://github.com/phanein/deepwalk.
[5] https://github.com/tangjianpku/LINE.
[6] https://github.com/sharadnandanwar/snbc.
[7] https://github.com/aditya-grover/node2vec.

Table 2. 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% nodes labeled 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	68.97	**72.25**
	Youtube	22.82	31.94	36.63	33.90	35.06	33.86	42.19	**44.03**
	CoRA	55.83	63.02	71.37	65.50	66.75	72.66	77.80	**77.95**
	IMDb	**33.59**	22.21	33.12	30.39	30.18	32.97	26.96	26.89
	Pubmed	50.32	65.68	77.40	68.31	79.22	79.02	80.13	**80.92**
	Wikipedia	45.10	65.29	71.10	68.812	68.78	70.69	**75.38**	73.58
	Flickr	21.37	29.67	28.73	30.96	24.20	30.65	28.22	**33.12**
	Blogcatalog	17.89	27.04	25.63	25.32	22.40	27.46	**31.87**	31.05
	PPI	6.28	8.61	8.14	9.27	7.97	8.88	19.80	**20.95**
	POS	23.05	21.06	31.40	38.24	37.73	34.59	43.92	**44.16**
	Average	31.00	37.28	41.53	41.12	43.12	43.99	49.52	**50.49**
Micro-F_1 (%)	Amazon	34.86	39.62	33.06	42.42	59.79	50.55	69.60	**73.04**
	Youtube	27.81	36.40	40.73	38.01	39.67	38.35	47.94	**49.17**
	CoRA	55.85	63.00	71.36	65.47	66.78	72.66	77.80	**77.96**
	IMDb	**42.62**	29.99	41.82	39.89	39.53	42.36	36.29	36.29
	Pubmed	50.32	65.68	77.40	68.31	79.22	79.02	80.13	**80.92**
	Wikipedia	48.51	66.95	72.19	70.21	70.68	72.07	**76.85**	75.25
	Flickr	25.40	32.91	31.66	34.03	27.60	33.76	31.86	**36.55**
	Blogcatalog	20.50	28.86	27.29	27.45	24.66	29.41	**34.25**	33.56
	PPI	18.41	12.29	11.52	13.16	11.32	12.80	24.92	**25.73**
	POS	26.04	24.42	35.98	42.70	41.99	39.09	47.70	**48.23**
	Average	35.03	40.01	44.30	44.16	46.12	47.00	52.73	**53.67**
Macro-F_1 (%)	Amazon	32.00	35.95	21.64	37.52	56.84	45.85	66.39	**70.64**
	Youtube	18.17	34.19	33.92	33.47	32.07	32.60	40.71	**42.59**
	CoRA	43.16	56.82	62.68	59.07	55.68	64.79	72.10	**72.20**
	IMDb	18.89	18.77	18.22	18.83	17.45	18.46	26.61	**27.06**
	Pubmed	41.57	64.85	75.92	66.66	77.16	77.50	71.20	**79.89**
	Wikipedia	45.58	58.93	62.29	62.17	61.99	64.90	**70.04**	69.39
	Flickr	15.54	18.28	17.13	**21.80**	7.36	18.46	13.71	21.56
	Blogcatalog	11.47	**18.88**	14.65	15.52	8.29	17.16	17.44	16.76
	PPI	7.35	10.59	9.61	10.82	8.27	11.27	21.67	**22.00**
	POS	3.91	6.05	8.26	8.93	5.92	8.61	13.49	**14.29**
	Average	23.76	32.33	32.43	33.47	33.10	35.96	41.33	**43.63**

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 [6] to return the most probable labels as described in prior work [20,23,27].

4.3 Performances of Classifiers

In this part, we study the performances of within-network classifiers in different datasets respectively. As some baseline algorithms are just designed for undirected or unweighted graphs, we just transform all the graphs to undirected and unweighted ones for a fair comparison.

First, to study the performance of different 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.

Table 2 shows the average Hamming score, Macro-F$_1$ score, and Micro-F$_1$ score for multi-label classification results in the datasets. Numbers in bold show the best algorithms in each metric of different datasets. As shown in the table, in most of the cases, IDRN and IDRN$_c$ algorithms improve the metrics over the existing baselines. For example, in the **Amazon** network, IDRN$_c$ outperforms all baselines by at least 22.46%, 22.16% and 24.28% with respect to Hamming score, Macro-F$_1$ score, and Micro-F$_1$ score respectively. Our model with community priors, i.e., IDRN$_c$ often performs better than IDRN with global prior. For the three metrics, IDRN and IDRN$_c$ perform consistently better than other algorithms in the 10 datasets except for **IMDb**, **Flickr** and **Blogcatalog**. 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 algorithm, 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 2. On average our approach can provide Hamming score, Micro-F$_1$ score and Macro-F$_1$ score up to 14%, 21% and 14% higher than competing methods, respectively. The results indicate that our IDRN with community priors outperforms almost all baseline methods when networks are sparsely labeled.

Second, we show the performances 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% to 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 3 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 2. As shown in Fig. 1, IDRN and IDRN$_c$ perform consistently better than other algorithms in these 3 datasets in Fig. 1. In fact, nearly in all the 10 datasets, our approaches outperform all the baseline methods significantly. When the networks are sparsely labeled (i.e., with 10% or 20% labeled data), IDRN$_c$ outperforms slightly better than IDRN. However, when more nodes are labeled,

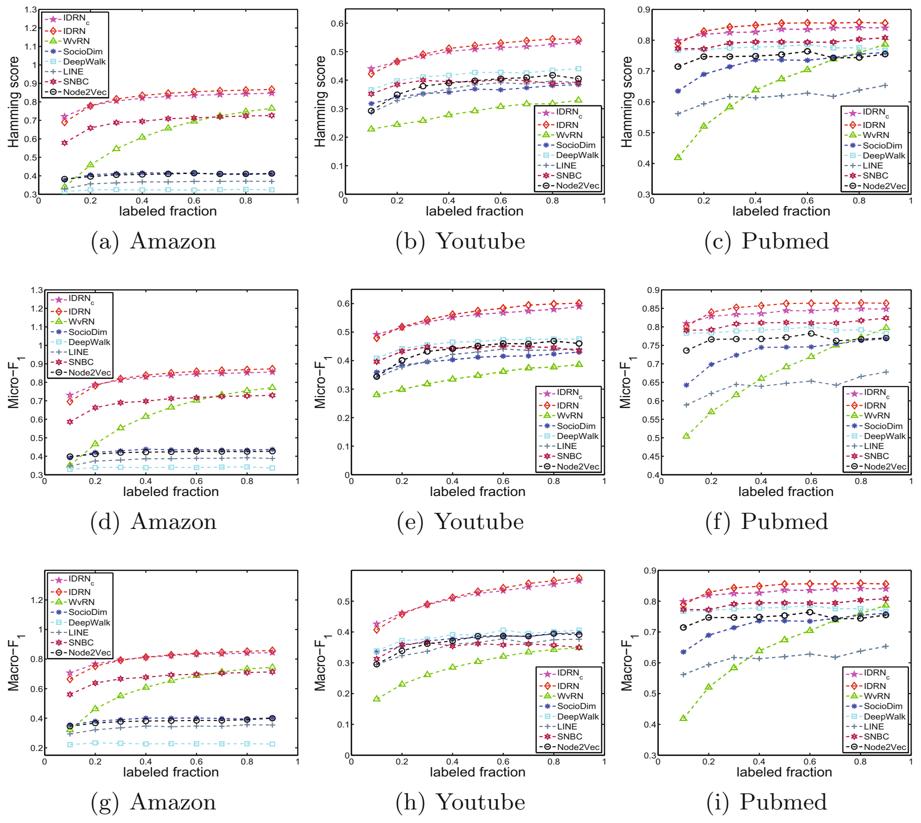

Fig. 1. Performance evaluation of Hamming scores, Micro-F$_1$ scores and Macro-F$_1$ scores on varying the amount of labeled data used for training. The x axis denotes the fraction of labeled data, and the y axis denotes the Hamming scores, Micro-F$_1$ scores and Macro-F$_1$ scores, respectively.

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 datasets are small, while the opposite conclusion holds for training on large datasets.

5 Conclusion and Discussion

In this paper, we propose a novel approach for node classification, which combines local node identifiers and community priors to solve the multi-label node classification problem. In the algorithm, we use the node identifiers in the egocentric networks as features and propose a within-network classification model by incorporating community structure information. 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 several publicly available datasets. When networks are sparsely labeled, on average our approach can provide Hamming score, Micro-F_1 score and Macro-F_1 score up to 14%, 21% and 14% higher than competing methods, respectively. Moreover, our method is quite practical and efficient, since it only requires the features extracted from the network structure without any extra data which makes it suitable for different real-world within-network classification tasks.

Acknowledgments. 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. Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 37–48 (2013)
2. Barabási, A.-L., Albert, R.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)
3. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. CoRR, abs/1101.3291 (2011)
4. Bian, J., Chang, Y.: 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, pp. 2425–2428 (2011)
5. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. **10**, 10008 (2008)
6. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
7. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
8. Fortunato, S., Hric, D.: Community detection in networks: a user guide. Phys. Rep. **659**, 1–44 (2016)
9. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. **99**(12), 7821–7826 (2002)
10. Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
11. Jiang, S., Hu, Y., et al.: 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, SIGIR 2016, pp. 185–194 (2016)
12. Joulin, A., Grave, E., et al.: Bag of tricks for efficient text classification. CoRR, abs/1607.01759 (2016)
13. Macskassy, S.A., Provost, F.: A simple relational classifier. In: Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003, pp. 64–76 (2003)

14. Macskassy, S.A., Provost, F.: Classification in networked data: a toolkit and a univariate case study. J. Mach. Learn. Res. **8**(May), 935–983 (2007)
15. Marsden, P.V.: Egocentric and sociocentric measures of network centrality. Soc. Netw. **24**(4), 407–422 (2002)
16. McDowell, L.K., Aha, D.W.: Labels or attributes? Rethinking the neighbors for collective classification in sparsely-labeled networks. In: International Conference on Information and Knowledge Management, pp. 847–852 (2013)
17. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR, abs/1301.3781 (2013)
18. Murphy, K.P., Learning, M.: A Probabilistic Perspective. The MIT Press, Cambridge (2012)
19. Nandanwar, S., Murty, M.N.: Structural neighborhood based classification of nodes in a network. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1085–1094 (2016)
20. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
21. Rayana, S., Akoglu, L.: Collective opinion spam detection: bridging review networks and metadata. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 985–994 (2015)
22. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077 (2015)
23. Tang, L., Liu, H.: Relational learning via latent social dimensions. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 817–826 (2009)
24. Tang, L., Liu, H.: Scalable learning of collective behavior based on sparse social dimensions. In: The 18th ACM Conference on Information and Knowledge Management, pp. 1107–1116 (2009)
25. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1225–1234 (2016)
26. Wang, S.I., Manning, C.D.: Baselines and bigrams: simple, good sentiment and topic classification. In: Proceedings of the ACL, pp. 90–94 (2012)
27. Wang, X., Sukthankar, G.: Multi-label relational neighbor classification using social context features. In: Proceedings of The 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 464–472 (2013)
28. Yin, D., Hu, Y., et al.: Ranking relevance in yahoo search. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 323–332 (2016)