Network Embedding with Deep Metric Learning

Xiaotao CHENG†(a), Member, Lixin JI†, Ruiyang HUANG†, and Ruifei CUI††, Nonmembers

SUMMARY Network embedding has attracted an increasing amount of attention in recent years due to its wide-ranging applications in graph mining tasks such as vertex classification, community detection, and network visualization. Network embedding is an important method to learn low-dimensional representations of vertices in networks, aiming to capture and preserve the network structure. Almost all the existing network embedding methods adopt the so-called Skip-gram model in Word2vec. However, as a bag-of-words model, the skip-gram model mainly utilized the local structure information. The lack of information metrics for vertices in global network leads to the mix of vertices with different labels in the new embedding space. To solve this problem, in this paper we propose a Network Representation Learning method with Deep Metric Learning, namely DML-NRL. By setting the initialized anchor vertices and adding the similarity measure in the training progress, the distance information between different labels of vertices in the network is integrated into the vertex representation, which improves the accuracy of network embedding algorithm effectively. We compare our method with baselines by applying them to the tasks of multi-label classification and data visualization of vertices. The experimental results show that our method outperforms the baselines in all three datasets, and the method has proved to be effective and robust.

key words: deep metric learning, network representation learning, likelihood label, anchor initialization, semi-supervised learning

1. Introduction

Entities in the real world usually interact with each other to form large-scale complex networks, such as social networks, biological networks, and citation networks. It is well recognized that network data are so sophisticated that makes it a challenging task to deal with. To process such data effectively, the first challenge is to find an effective network representation, that is, to represent networks concisely so that analytic tasks can be conducted efficiently in both time and space. Traditionally, we represent a network with an adjacency matrix. For large networks with billions of nodes, the traditional network representation poses several challenges to network processing and analysis. The high computational complexity, low parallelizability, and inapplicability of machine learning methods make it infeasible for large-scale real-world networks. One promising strategy is to learn a vector representation of an information network [1]–[3]. Each network vertex is represented as a low-dimensional vector so that the information conveyed by the original network can be effectively captured. As a result, existing machine learning methods can be directly applied in the low-dimensional vector space to conduct network analytic tasks such as vertex classification, network clustering, etc.

In the past few years, a series of algorithms have been proposed for the Network Representation Learning (NRL), such as DeepWalk [4], LINE [5], Node2vec [6], and SEANO [7]. These algorithms have been shown to be effective in a variety of network analytic tasks, e.g., node classification, link prediction, community discovery, etc. However, most of them only considered the network structure, e.g., the links between nodes, whereas they ignored other label information that could benefit network representation learning and subsequent analytic tasks. In many applications, it usually has knowledge about the labels of some vertices in an information network. For example, in a social network such as Facebook or Twitter, there contain groups [8] or community information [9] for some users. In a citation network, we may know the topics that the documents belong to. We address the problem of semi-supervised embedding in information networks by leveraging the labels. A work similar to ours is Planetoid [10], which proposes a semi-supervised learning method based on graph embedding. Those methods generate the vertices sequence by random walk, and then use the Skip-gram model of the neural probabilistic language model Word2vec [11], [12] to generate the representations of the network vertices. However, the Skip-gram model is used as a bag of words model in the sliding window. There is a lack of description of the global information in the network. Due to the inter-class similarity and intra-class difference of vertices, it leads to the problem that vertices are mixed with each other in the new embedding space which is difficult to distinguish.

To solve the above-mentioned difficulties, we propose a new algorithm called Network Representation Learning with Deep Metric Learning (DML-NRL), which combines known label information and the network structure to jointly learn the network representations. It was inspired by the deep metric learning in image processing [13], [14]. In contrast to existing approaches in NRL, metric learning becomes a very appealing technique because of its ability to learn the general concept of distance metrics and its compatibility with efficient nearest neighbor inference on the learned new embedding space. Those embeddings trained by metric
learning are optimized to pull examples with different class labels apart from each other and push examples from the same classes closer to each other. One of the main advantages of our scheme is that the network jointly learns vertex representation from the aspects of structural similarity and label similarity. The anchor initialization and metric learning process in our scheme can separate the vertices with different labels from each other, and the embedding process based on the Skip-gram model ensures that vertices with similar structures are adjacent to each other. By controlling the proportion of two kinds of loss functions, we obtain a better network embedding. Empirically, we conduct the experiments on three real-world network datasets. Compared with baselines, the results show that the representations generated by our method can achieve better performance on node multi-label classification and data visualization tasks.

In summary, the contributions of this paper are listed as follows:

- We introduce deep metric learning into the generation of network embedding, and also build better representations that are suitable for multi-label classification. This method can jointly capture information about the network structure and available labels.
- We propose an effective strategy to get the vertex’ maximum likelihood labels, which can be useful in the presence of label sparsity. By setting the anchor vertices in the process of vector initialization, the representations of network embedding have improved.
- We empirically evaluate the effectiveness of the proposed algorithm through vertex multi-label classification and data visualization tasks on real-world citation networks, showing its superior performance over the state-of-the-art baselines. We also demonstrate the flexibility of our algorithm by adjusting the ratio of metric learning.

## 2. Related Work

Network representation learning enjoys increasing popularity in recent years. In this section, we review two categories of network embedding algorithms. It is network structure and property preserving network embedding methods that are based on network structure only, and side information preserving network embedding method that combining vertex content with network structure to enhance network embedding.

Most of the research on network representation learning is based on network structure information. DeepWalk [4] algorithm is one of the pioneer works for learning node representations in networks. Following the idea of the Skip-gram [12] model, DeepWalk generates node context using truncated random walks and learns node representations. LINE [5] formulates a more clearer objective function to preserve the first-order proximity and the second-order proximity. Node2vec [6] is an improved version of DeepWalk. To capture both the local and global network structure, node2vec proposes a balanced strategy between the BFS (Breadth-First-Search) and DFS (Depth-First-Search). It exploits biased random walks to generate context nodes, and then applies DeepWalk to learning node representations. Recently, SDNE is proposed to learn non-linear network representations by applying deep auto-encoder model on node adjacency matrix and exploiting the first-order proximity as supervised information. The above NRL algorithms only consider network structure without taking other user-generated content available in information networks to learn more informative network representations.

However, the real-world network vertices are often accompanied with rich side information, such as node content or labels in information networks [15], [16], node and edge attributes in social networks [17], as well as node types in heterogeneous networks [18], [19]. Side information provides useful clues for characterizing relationships among network nodes, and thus is helpful in learning node embedding representation. When the network structure is relatively sparse, the importance of the side information as complementary information source is increasingly outstanding. In order to integrate and balance the structure and side information in network embedding. Some multimodal and multi-source fusion techniques has been studied. Chen et al. [8] proposed a model GENE, which means group enhanced network embedding. By randomly sampling a group label and assign it to vertex sequences for training, it achieves the fusion of network structure information and label information. Yang et al. [10] proposed semi-supervised learning method of graph embedding. There are two types of context that are sampled in this algorithm. The first type of context is based on the graph, which encodes the structure information, and the second type of context is based on the labels, which we use to inject label information into the embedding. TriDNR [20] further exploits supervised labels to learn better node representations by modeling the inter-node relationship, node-word correlation and label-word correspondence. In recent years, network embedding methods has developed from the general representation of network to task-driven representation, aimed at improving the performance of specific applications [21]–[24].

Metric learning aims at learning semantic distance measures and embeddings such that similar input vertices are mapped to nearby points on a manifold and dissimilar vertex are mapped apart from each other. We briefly review recent works on training neural network to learn semantic embeddings in deep metric learning. Contrastive embedding [25] is trained on the paired data. Intuitively, the contrastive training minimizes the distance between a pair of examples with the same class label and penalizes the negative pair distances for being smaller than the margin parameter $\alpha$. Triplet embedding [26], [27] is trained on the triplet data $\{(x_a, x_p, x_n)\}$ where $(x_a, x_p)$ have the same class labels and $(x_a, x_n)$ have different class labels. The $x_a$ term is referred to as an anchor of a triplet. Intuitively, the training process encourages the network to find an embedding where
the distance between $x_{i}^{(1)}$ and $x_{j}^{(1)}$ is larger than the distance between $x_{i}^{(1)}$ and $x_{j}^{(1)}$ plus the margin parameter $\alpha$. In this paper, we define a structured loss function based on multiple positive and negative pairs of samples in the training set, and use partial labeled nodes referred to as anchor points to improve the performance of network representation.

3. Network Embedding with Deep Metric Learning

In this section, we first define the problem. Then we introduce the proposed semi-supervised network representation learning model of DML-NRL. At last, we present some discussions and analysis of the model.

3.1 Problem Definition

**DEFINITION 1. Information Network [15].** Information network is an undirected graph $G = (V, E, X, Y)$, where $V = (v_1, v_2, \ldots, v_n)$ is the set of vertices, $E$ is the set of edges, $X = (x_1, x_2, \ldots, x_n)$ is the attribute information matrix, and $Y = (y_1, y_2, \ldots, y_n)$ are the labels of the vertices.

**DEFINITION 2. Network Embedding [2].** It is also known as network representation learning. Given a network denoted by $G = (V, E)$, network embedding aims to learn a mapping function $f: v_i \rightarrow e_i \in \mathbb{R}^d$, where $d \ll |V|$. The objective of the function is to make the similarity between $e_i$ and $e_j$ explicitly preserve the similar topological structure information and attribute information of $v_i$ and $v_j$.

**DEFINITION 3. Semi-supervised Embedding in Information Network.** Given the information network with partial labels, we aim to learn a robust low-dimensional vector representation $e_i \in \mathbb{R}^d$ for each vertex $v_i$, where $d \ll |V|$. $e_i$ can jointly capture the information of the attributes, the graph structure, and the partial labels in information network. This paper focuses on the method of how to improve the performance of semi-supervised embedding on information networks.

We define some of the terms and notations in Table 1 which will be used later.

3.2 Model Framework

The network embedding method in this paper should meet the following two conditions: Firstly, it should maintain the adjacent relationship of vertices in the original network; Secondly, the same kind of vertices should be adjacent to each other while different types of vertices should be far away after the network is embedded. In the scheme of network embedding model incorporating deep metric learning, we design two kinds of loss functions to make the network training meet the above requirements. That is, the unsupervised embedding loss and supervised metric loss. By comparing and measuring similarities between pairs of examples, the model adjusts the gradient descent direction of vertices embeddings. So that the same categories of vertices are adjacent to each other while different categories of vertices are far away from each other, and the optimization of network representation learning algorithm is realized. The model architecture is illustrated in Fig. 1.

The model we purposed mainly consists of three parts. Firstly, it generates the vertex sequence by using random walks on a network, similar to the existing algorithms (such as DeepWalk, Node2vec and its variants). Then the vertex sequence will be trained to generating vector representation for each vertex using Skip-gram model in Word2vec. Initialization process means assigning the initial values of the model parameters. In the initialization process of vertex vectors, the initialization vectors of vertices with the same label are adjacent, while the initialization vectors with different label are far away. The update process is to continuously adjust the node representation vector until the model training is optimal. These partial labeled vertices are used as anchor points for metric learning. Finally, we calculate the unsupervised embedding loss when generating the network embedding and the supervised metric loss between vertices and anchor points. The training of network embedding jointly learns the gradients of two kinds of losses. Therefore, by controlling the proportion of two kinds of losses in joint training, we can maximize the likelihood of random walk sequences while ensuring the distribution of similar classes of vertices in the embedding space.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>Partially labeled attributed network</td>
</tr>
<tr>
<td>$V$</td>
<td>Vertex set</td>
</tr>
<tr>
<td>$E$</td>
<td>Edge set</td>
</tr>
<tr>
<td>$X$</td>
<td>Attribute information matrix</td>
</tr>
<tr>
<td>$Y$</td>
<td>Vertex labels matrix</td>
</tr>
<tr>
<td>$\mathbb{R}^d$</td>
<td>Matrix of network embedding</td>
</tr>
<tr>
<td>$d$</td>
<td>Dimension of network embedding</td>
</tr>
<tr>
<td>$e_i$</td>
<td>Embedding of vertex $i$</td>
</tr>
<tr>
<td>$L_{m}$</td>
<td>Loss function of metric learning</td>
</tr>
<tr>
<td>$L_{e}$</td>
<td>Loss function of embedding learning</td>
</tr>
</tbody>
</table>

![Fig. 1](image-url)
(1) Anchor Initialization and Vertex Label Annotation

**Anchor initialization:** According to whether the vertex labels are known, we divide the vertices into labeled vertices and unlabeled vertices. In this paper, anchor initialization means that we select some labeled nodes as the initialization center of vertices with the corresponding labels, and they are the anchors of this class. The initialization vectors with the same labels are adjacent in the new embedding space.

**Vertex label annotation:** In this paper, the partially labeled information network only has a small number of labeled vertices. So, we need vertex label annotation to assign a likelihood label to other unlabeled vertices. Label annotation means adding a label to a node that it might belong to. According to the hypothesis [28], labels of adjacent vertices are more likely to be similar and vertices which have similar neighborhoods acquire similar representations. At the random walk paths with different node $v_i$ as the root node, we record the labeled vertex information that this vertex first encountered in the path as the temp label of $v_i$. The temp label appearing most in the K times is the vertex’s maximum likelihood label $\hat{\gamma}_i$. K is the number of random walks. If the network structure is not tight enough, there are no labeled vertices on the random walk path. Then the maximum likelihood label $\hat{\gamma}_i$ of this vertex is empty.

In the d-dimensional embedding space, when the anchor initialization and vertex maximum likelihood label annotation are done, we can build vertex pairs to take similarity measurement. Section (2) in the paper below discusses the generation method of node embedding representation based on Skip-gram model. Section (3) defines the distance and similarity measure of the embedded representations of vertices.

(2) Unsupervised Embedding Loss

In this paper, the Skip-gram model in Word2vec is used in the generation of vertex embedded representation. Skip-gram is a language model that maximizes the co-occurrence probability of a word appearing within a window in a sentence. So the Skip-gram model can capture the relationship between the target node and the context node. For each target vertex $v_i$, the context vertices sequence within the sliding window $w$ is $W_{v_i} = \{v_{i-w}, v_{i-w+1}, \ldots, v_{i+w}\}$. Given the representation of $v_i$, we would like to maximize the probability of its neighbors in the walk. In terms of vertex representation modeling, this yields the optimization problem:

$$\min - \log \Pr(v_{i-w}, \ldots, v_{i+w} \mid v_i | \Phi(v_i)). \quad (1)$$

It approximates the conditional probability in Eq. (1) using an independence assumption as the following,

$$\Pr(v_{i-w}, \ldots, v_{i+w} \mid v_i | \Phi(v_i)) = \prod_{j=1-w, j \neq i}^{i+w} \Pr(\Phi(v_i)). \quad (2)$$

The unsupervised embedding loss function of all the vertices in the network then reads,

$$L_u = -\sum_{v_i \in V} \log p(v_{i-w}, v_{i-w+1}, \ldots, v_{i+w} \mid v_i)$$

where $p(v_{i-w}, v_{i-w+1}, \ldots, v_{i+w} \mid v_i)$ is the likelihood of observing the vertex $v_{i+w}$ in the target context given the target vertex $v_i$. The $p(v_{i-w}, v_{i-w+1}, \ldots, v_{i+w} \mid v_i)$ can be computed using the soft-max function as follows:

$$p(v_{i-w}, v_{i-w+1}, v_{i-w+2} \mid v_i) = \frac{\exp(e^Tv_{i-w} \cdot e^Tv_{i-w+1} \cdot e^Tv_{i-w+2})}{\sum_{v_i \in V} \exp(e^Tv_{i-w} \cdot e^Tv_{i-w+1} \cdot e^Tv_{i-w+2})}, \quad (4)$$

where $e^Tv_{i-w}$ and $e^Tv_{i-w+1}$ are the input and output vector representation of vertex $v$.

The gradient calculation method for the above embedding loss has a more detailed derivation in [29]. It uses the way of constructing Huffman tree to calculate the representations of vertices, and no more derivation is given here.

(3) Supervised Metric Loss

In order to make full use of some known label information, we introduce supervised metric learning method after unsupervised embedding training. It can avoid the bias of network representations that only focus on the unsupervised embedded representations learning by using the metric learning of labeled vertices. Suppose that the network embedding of the node $v_i$ is $e_i$ and the embedding of $v_j$ is $e_j$, then the Euclidean distance of the node pair $(i, j)$ in the d-dimensional embedding space is

$$D_{ij} = \|f(v_i) - f(v_j)\|_2 = ||e_i - e_j||_2, \quad (5)$$

where $f(\cdot)$ is the mapping function representing $e_i$ from node $v_i$ to its embedding.

In order to realize the normalization of distance measure and avoid the convergence caused by the scale difference in the joint loss function, we define the similarity between vertex pairs. Given the embedded representations of vertices $v_i$ and $v_j$, the similarity between the vertex $v_i$ and $v_j$ is defined as follows,

$$\sigma(i, j) = \frac{2}{1 + \exp(D_{ij})} = \frac{2}{1 + \exp(||e_i - e_j||_2)}. \quad (6)$$

We can see that when vertices are close to each other in the embedding space, there is $\sigma(i, j) = 2/(1 + e^0) = 1$. When the vertices are far away in the embedding space, there is $\sigma(i, j) = 2/(1 + e^\infty) = 0$.

According to the partially labeled vertices information, we aim to get the optimal representation of network vertices. Minimizing the intra class spacing and maximizing the inter class spacing is the problem that we need to focus on. The similarity between two vertices is determined by the label information. The embedding representations of vertices are affected by the distance between two nodes. We define a structured loss function based on multiple positive and negative pairs of samples in the training set. The network is trained by minimizing the loss function in Eq. (7),

$$L_e = -\frac{1}{|S|} \sum_{v_i \in V, v_i \notin S} \sum_{v_j \in S, y_j \in C} \sum \{I[y_i = y_j] \log(\sigma(i, j)) \}$$
method in this paper is to minimize the joint loss function of the multi-layer neural network based on the Skipgram model in Word2vec. As shown in Eq. (10), it includes two parts: the unsupervised embedding loss that generates the vertex embedded representation and the supervised measurement loss that represents the similarity between the intra class dispersion and the inner class similarity. The specific optimization variable is the constantly updated vertices representation in the neural network training. The loss function is as follows,

$$L = \lambda_1 L_u + \lambda_2 L_c.$$  

where $L$ is the total loss of network training, $L_u$ is the embedding loss by maximizing the random walk sequence, as shown in Eq. (3), $L_c$ is the distance metric loss, as shown in Eq. (7). $\lambda_1$ and $\lambda_2$ are the weight balance coefficient of two kind of loss. By adjusting the importance of two losses in the training stage, it can avoid the bias of network representations that only focus on one side. In the experiment, we achieve the optimization of network embedding representation $e_i$ by loss back propagation algorithm. This paper will discuss the effects of $\lambda_1$ and $\lambda_2$ on the experimental results of network node classification in the experimental section. Our model contains two main stages, sampling and training. We will then illustrate the two steps in details.

### 3.3 Vertex Sampling Process

Vertex sampling is the process of random walk on the network. In the sampling stage, most methods uniformly take a random vertex $v_i$ as the root of a random walk $W_{ni}$, then from the root they sample the neighbors of the last vertex visited until they reach the maximum length $L$. After this step, they can get $W_{ni} = \{v_i, v_{i+1}, v_{i+2}, \ldots, v_{in}\}$, where $v_{in}$ is one of the neighbors of $v_{i+n-1}$. Our method is used as the lifting method of the existing network representation algorithm. The walking strategy in the literature [4], [6] can still be adopted in the generation of random walk sequence of vertices. At the same time, we add maximum likelihood labels to unlabeled vertices based on the hypothesis that vertices which have similar neighborhoods acquire the same label. Thus it enhances the usage of small number of labeled vertices in the network, and makes the metric learning applies on vertex pairs possible.

#### Label Annotation in Random Walk:

In the process of random walk with each vertex as the root node, the first label information that the vertex encountered is recorded as the temp label of this unlabeled vertex, and its maximum likelihood label is determined by the most appearance of the temp label. If there are no labeled vertices on the random walk path, then the maximum likelihood label of this vertex is empty. The training of metric learning is based on the likelihood label set by label annotation. The vertex sampling process is shown in Algorithm 1.
3.4 Model Training Process

The model training process is to minimize the loss function in Eq. (10). In the joint training process, the stochastic gradient descent method is used to add the gradient of embedding loss and the metric learning loss. Then it propagates back to the rest of the network. For the problem of large amount of computation for all vertices in Eq. (3), the method of negative sampling [6] is used to approximate calculation. The model training process is shown in Algorithm 2.

3.5 Complexity Analysis

In this paper, we mainly improve the vertices sampling and training process of the existing algorithms. In vertices sampling process, the statistics of unlabeled vertices’ likelihood label is based on the steps of generating random walk vertex sets. The likelihood label of unlabeled vertices can be obtained by setting the mark information during the random walk, and it does not improve the algorithm complexity of the vertices sampling process.

In the model training process, the random gradient descent method which is suitable for large-scale data processing is used. The gradient calculation in Eq. (4) is complex, requiring traverses all vertices in the network. This paper adopts Hierarchical soft-max method in [6]. This reduces the computational complexity of calculating Eq. (6) from $O(|V|^2)$ to $O(|V| \log(|V|))$, where $|V|$ is the number of vertices in the network. The complexity of metric learning in Eq. (7) is determined by the similarity calculation between the pair of network vertex and selected anchors. It is $O(|S| \cdot (|V| - |S|))$. For the number of selected labeled vertices $|S|$ are less than $|V|$, and $|S| = \alpha |V|$, $0 < \alpha < 1$, where $\alpha$ is the ratio of anchor vertices. The overall complexity of our algorithm is between $O(|V| \log(|V|))$ and $O(|V|^2)$, which is dominated by the former term.

4. Simulation Study

4.1 Evaluation Metrics

In the evaluation of network representation learning algorithms, nodes multi-label classification, link prediction, and visualization tasks are generally used. In this paper, we choose multi-label classification and visualization task as done in [4], [17], [20]. Label classification is an important application in social network analysis. Since the label information is rather sparse, manual annotation is often needed.

For the multi-label classification task, we adopt Macro-$F_1$ and Micro-$F_1$ as many other works do. Specifically, for a label $A$, we use $TP(A)$, $FP(A)$ and $FN(A)$ to denote the number of true positives, false positives and false negatives in the instances which are predicted as $A$, respectively.

Suppose $C$ is the overall label set. The related evaluating indicators are defined as follows:

$$\text{precision} = \frac{TP(A)}{TP(A) + FP(A)}$$

$$\text{recall} = \frac{TP(A)}{TP(A) + FN(A)}$$

$$F_1(A) = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{Macro} - F_1 = \frac{\sum_{A \in C} F_1(A)}{|C|},$$

where $F_1(A)$ is the $F_1$-measure for the label $A$. Macro-$F_1$ is a metric which gives equal weight to each class. Micro-$F_1$ is a metric which gives equal weight to each instance. It is defined as follows,

$$Pr = \frac{\sum_{A \in C} TP(A)}{\sum_{A \in C} (TP(A) + FP(A))}$$

$$R = \frac{\sum_{A \in C} TP(A)}{\sum_{A \in C} (TP(A) + FN(A))}$$

$$\text{Micro} - F_1 = \frac{2 \cdot Pr \cdot R}{Pr + R}.$$
we use three citation network datasets. The detailed descriptions are listed as follows, and the detailed statistics of the datasets are summarized in Table 2.

- **CiteSeer-M10**† data [26]. This data set contains 10 distinct science journals. Agriculture, Archaeology, Biology, Computer science, Financial economics, Industrial engineering, Science, Material, Petroleum chemistry, Physics, Social science.
- **Cora**†† data. It is a collection of papers selected from machine learning directions that contains 7 subclasses.
- **DBLP**††† data. Citation network data in the field of computer science, the data used in this paper are divided into 4 fields. Database, Data Mining, Artificial Intelligent, Computer Vision. It contains 60744 articles, 52890 edges.

In this paper, we take the following five methods as the baselines.

- **Doc2vec**[11]: It is the Paragraph Vectors algorithm which embeds any piece of text in a distributed vector using neural network models. Here we use the attribute information of the paper in the citation network to generate the vector representation. In this experiment, we mainly use it as a contrast of other algorithms using network structure information.
- **DeepWalk**[4]: It employs truncated random walks on the plain graph and involves language modeling techniques, i.e., Word2vec, to analyze the walking tracks. The embedding representation of network vertices is generated by using the network structure only.
- **Node2vec**[6]: Node2vec is an improved version of DeepWalk. According to the balanced strategy between the given depth first and breadth first, the biased random walk and the Skip-gram model are used to generate the embedded representation of the network nodes.
- **GENE**[8]: It uses a group vector to predict the vertices randomly sampled in the same group, in which vertices in that group are placed closely. The group vector acts as a memory cell that contains both the information of vertices in that group and across groups, resulting in a closer distance between some vertices in the same groups. It enhances the network representation by using group label information in the training process.
- **TriDNR**[20]: It is a deep network representation algorithm that exploits network structure, node content, and label information for learning. In order to compare with other algorithms, we only use node structure and node label information in this algorithm.

- **DML-NRL**: It is our proposed method. The algorithm combines the embedded representation with metric learning to achieve similar clustering, and distinguishes the obvious goals between different classes.

The parameter settings for baselines are as follows. We set embedded dimension to $d = 300$, sliding window size to $w = 5$, walk length to $walk\_length = 16$, walk times to $num\_walks = 10$. For the Node2vec algorithm, the random walk parameter is $p = 2, q = 1$.

### 4.3 Experiment Results

We evaluate the effectiveness of different network representations through a multi-label classification task in this experiment. The representations for the vertices are generated from the network embedding methods and are used as features to classify each vertex into a set of labels. We repeat the experiment five times and report the averaged Macro-$F_1$ and Micro-$F_1$. The results are shown in Figs. 3 and 4, respectively. The labeled vertices ratio is 0.1. The histogram of our method is consistently higher than the baselines. It demonstrates that, compared with the baselines, the learned network representations of our method can be better generalized to the classification task.

In Tables 3 and 4, the average accuracy of node classification under different label ratios on the same data set is shown. In Tables 4 and 5, the average accuracy (Micro-$F_1$...
value) of node classification on different datasets under the same label ratios is shown.

From the above results, we have the following observations and analysis.

It can be seen that the accuracy improves with the training size increasing from 10% to 90%. Compared with DeepWalk and Node2vec, our method has a great improvement. The reason is that DeepWalk and Node2vec are unsupervised algorithm. DML-NRL makes full use of partial label information to realize semi-supervised embedding representation. The experimental results also show that a small number of labeled vertices can improve the classification performance obviously. In real-world applications, the performance of classification can be greatly improved by manual labeling a small number of samples. The important role of label information is also proved.

Compared with GENE and TriDNR which also use the label information to realize semi-supervised embedding representation, our method is more significantly effective in using label information. The GENE algorithm randomly sample a group label from the groups \( v_i \) belongs to and assign it to node sequence. Those node sequences with group label are then used for training process. By applying the Skip-gram model to the node sequences, it indirectly affects the final embedded representation of vertices. The TriDNR algorithm also uses the interaction between the label information and the network structure information to carry out parameter transfer to optimize the embedded representation. However, our method uses similarity metric learning directly in the training process. By narrowing the distance between the same classes and increasing the distance between different classes, the utilization of label information is more direct and effective, which improves the effect of subsequent classification. At the same time, compared with the Doc2vec algorithm, we can see that the attribute information of nodes also plays an important role in the classification. Adding attribute information in the process of embedded representation can enhance the performance of embedded representation.

The loss function of the network embedding algorithm DML-NRL consists of two parts: metric learning loss and embedded loss. By adjusting the parameters \( \lambda_1 \) and \( \lambda_2 \), we can change the weight of the two kinds of losses in the embedding result. Since the main task of this paper is to learn the embedding representation of network, we set \( \lambda_1 \) equal to 1 and adjust \( \lambda_2 \). Table 6 shows the influence of classification with different parameters. When \( \lambda_2 \) is zero, there is no metric learning in the algorithm. When \( \lambda_2 \) is 0.3, the Micro-F1 of classification is the highest. As reflected in Table 6, we can see that the classification results of the metric learning is not obvious when \( \lambda_2 \) is small. With the increase of \( \lambda_2 \), the classification performance is slightly improved. When \( \lambda_2 \) is

### Table 3 Micro-F1 on CiteSeer-M10 on multi-label classification (%) (label = 0.01)

<table>
<thead>
<tr>
<th>TrainSize</th>
<th>Doc2Vec</th>
<th>Deepwalk</th>
<th>Node2vec</th>
<th>GENE</th>
<th>TriDNR</th>
<th>DML-NRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>61.01</td>
<td>66.27</td>
<td>66.27</td>
<td>66.27</td>
<td>66.27</td>
<td>66.27</td>
</tr>
<tr>
<td>30%</td>
<td>68.06</td>
<td>74.68</td>
<td>74.68</td>
<td>74.68</td>
<td>74.68</td>
<td>74.68</td>
</tr>
<tr>
<td>50%</td>
<td>69.50</td>
<td>77.56</td>
<td>77.56</td>
<td>77.56</td>
<td>77.56</td>
<td>77.56</td>
</tr>
<tr>
<td>70%</td>
<td>71.68</td>
<td>80.28</td>
<td>80.28</td>
<td>80.28</td>
<td>80.28</td>
<td>80.28</td>
</tr>
<tr>
<td>90%</td>
<td>73.52</td>
<td>82.16</td>
<td>82.16</td>
<td>82.16</td>
<td>82.16</td>
<td>82.16</td>
</tr>
</tbody>
</table>

### Table 4 Micro-F1 on CiteSeer-M10 on multi-label classification (%) (label = 0.2)

<table>
<thead>
<tr>
<th>TrainSize</th>
<th>Doc2Vec</th>
<th>Deepwalk</th>
<th>Node2vec</th>
<th>GENE</th>
<th>TriDNR</th>
<th>DML-NRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>69.03</td>
<td>76.37</td>
<td>76.37</td>
<td>76.37</td>
<td>76.37</td>
<td>76.37</td>
</tr>
<tr>
<td>30%</td>
<td>71.87</td>
<td>83.46</td>
<td>83.46</td>
<td>83.46</td>
<td>83.46</td>
<td>83.46</td>
</tr>
<tr>
<td>50%</td>
<td>73.69</td>
<td>85.62</td>
<td>85.62</td>
<td>85.62</td>
<td>85.62</td>
<td>85.62</td>
</tr>
<tr>
<td>70%</td>
<td>75.88</td>
<td>87.56</td>
<td>87.56</td>
<td>87.56</td>
<td>87.56</td>
<td>87.56</td>
</tr>
<tr>
<td>90%</td>
<td>77.98</td>
<td>89.32</td>
<td>89.32</td>
<td>89.32</td>
<td>89.32</td>
<td>89.32</td>
</tr>
</tbody>
</table>

### Table 5 Micro-F1 on Cora on multi-label classification (%) (label = 0.2)

<table>
<thead>
<tr>
<th>TrainSize</th>
<th>Doc2Vec</th>
<th>Deepwalk</th>
<th>Node2vec</th>
<th>GENE</th>
<th>TriDNR</th>
<th>DML-NRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>69.54</td>
<td>81.26</td>
<td>81.26</td>
<td>81.26</td>
<td>81.26</td>
<td>81.26</td>
</tr>
<tr>
<td>30%</td>
<td>72.22</td>
<td>84.18</td>
<td>84.18</td>
<td>84.18</td>
<td>84.18</td>
<td>84.18</td>
</tr>
<tr>
<td>50%</td>
<td>74.86</td>
<td>86.39</td>
<td>86.39</td>
<td>86.39</td>
<td>86.39</td>
<td>86.39</td>
</tr>
<tr>
<td>70%</td>
<td>79.36</td>
<td>87.85</td>
<td>87.85</td>
<td>87.85</td>
<td>87.85</td>
<td>87.85</td>
</tr>
<tr>
<td>90%</td>
<td>82.88</td>
<td>87.85</td>
<td>87.85</td>
<td>87.85</td>
<td>87.85</td>
<td>87.85</td>
</tr>
</tbody>
</table>

### Table 6 The influence of \( \lambda_2 \) on Cora when \( \lambda_1 \) is set to 1

<table>
<thead>
<tr>
<th>( \lambda_2 )</th>
<th>0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro-F1 (%)</td>
<td>75.89</td>
<td>77.21</td>
<td>79.86</td>
<td>83.72</td>
<td>81.56</td>
</tr>
</tbody>
</table>
large, the high weight of metric learning reduces the influence of context information in the sliding window, resulting in lower accuracy of vertex label prediction. Therefore, $\lambda_2$ is set to 0.3 in the experiment of this paper.

As shown in Figs. 5 and 6, the algorithm is relatively stable in different embedding dimensions. With the embedding dimension increasing, the performance of our algorithm tends to be relatively stable and robust.

Another important application for network embedding is to generate visualizations of a network on a two-dimensional space. Therefore we visualize the learned representations of network dataset. We use the low dimensional network representations learned by different embedding methods as the input to the visualization tool t-SNE [30]. As a result, each document is mapped as a two dimensional vector. Then we can visualize each vector as a point on a two dimensional space. For documents which are labeled as different categories, we use different colors for the corresponded points. Therefore, a good visualization result is that the points of the same color are close to each other. The visualization result is shown in Fig. 7. We can see that the results of Node2vec and GENE are not satisfactory because the points belonging to different categories are mixed with each other. For TriDNR, it is better than the former, but it is still mixed with each other. For DML-NRL, the result looks better because points with the same color form somewhat isolated groups. Obviously, the visualization of our method performs best in the aspects of group separation. The anchor initialization and metric learning can separate the vertices of different categories from each other, and the embedding progress based on Skip-gram model ensure that nodes with similar structures are close to each other. By controlling the proportion of two kinds of loss functions, we obtain better embedding representations.

5. Conclusion

In this paper, we proposed a Network Representation Learning method incorporating with Deep Metric Learning, namely DML-NRL. Specifically, to address the problem of vertices mixed with each other in the embedding space due to the inter-class similarity and intra-class difference, we set anchor vertices in network initialization and introduce the deep metric learning at the process of generating the network representations. It realized the combination of local structure information and global vertices’ label information. At the same time, we changed the weight of different losses in the representation by adjusting the parameter. Empirically, we evaluated the generated network representations in a variety of network datasets and applications. The results demonstrated that our method outperforms the baselines. It can be used as an effective performance optimization and supplements for the existing network representation algorithms based on random walk mode.

In the next step, we will focus on the conflict between network structure and side information in the network embedding. All the existing methods assume that there is an agreement between network structure and side information. To what extent the assumption holds in real applications, however, remains an open question.

Acknowledgments

We acknowledge Hongxin Zhi and Zhengming Liu for their inspirations. This work was partially supported by the
References


Xiaotao Cheng is a Ph.D. candidate at the China National Digital Switching System Engineering and Technological Research and Development Center. He received his bachelor degree from Nanjing University of Posts and Telecommunications in 2012. His research interests include network big data analysis and social network data mining.
Lixin Ji is a professor of China National Digital Switching System Engineering and Technological Research and Development Center. His research interests include information security of telecommunication network and network architecture.

Ruiyang Huang is an associate professor of China National Digital Switching System Engineering and Technological Research and Development Center. His research interests include network bigdata analysis and social network data mining.

Ruifei Cui is a Ph.D. candidate of the Radboud University Nijmegen. His research interest include bayesian learning and causal reasoning.