IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 1, 2018, accepted October 21, 2018, date of publication October 31, 2018, date of current version November 30, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2878870

MEMN:Multiple Vectors Embedding for Multi-Label Networks

JUHUA PU[1,2], ZHUANG LIU[1,2], YUJUN CHEN[1,2], AND XINGWU LIU[3,4]

[1]The State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
[2]Research Institute, Beihang University, Shenzhen 518057, China
[3]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[4]University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Xingwu Liu (liuxingwu@ict.ac.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1002000, in part by the National Natural Science Foundation of China under Grant 61502320, in part by the Science Foundation of Shenzhen City in China, and in part by the State Key Laboratory of Software Development Environment under Grant SKLSDE-2018ZX.

ABSTRACT Network embedding, which assigns vectors to network nodes in a manner that preserves the network features, is a hotspot of network research in recent years. A salient common feature of the existing approaches is that each node is mapped to exactly one vector. This one-vector mapping is insufficient to represent the nodes' attribution in those extensively existed networks whose nodes' have multiple labels. In this paper, we present MEMN, a novel approach of multiple vectors embedding for multi-labeled networks. For any node in the network, MEMN employs Node2vecWalk to generate its neighbor nodes. We maintain a neighbor cluster center for each label of the node and induce its label by clustering the embeddings of the neighbor nodes. Then, we assign vectors, one per label, to the node. This method can be non-parameterized, namely, NP-MEMN method. That is, if the number of label vectors for a node is not given, NP-MEMN can learn during embedding. Empirical studies on real datasets show that either MEMN or NP-MEMN outperforms many widely used methods in both multi-label classification and link prediction.

INDEX TERMS Network embedding, multiple vectors, multi-label classification, link prediction.

I. INTRODUCTION

Nowadays, information networks are becoming ubiquitous in a large spectrum of real-world applications in forms of social networks [1], biological networks [2], and citation networks [3], etc. Often, a discrete adjacency matrix is used to represent a network, which only captures vertices neighboring information. Such algorithms as [4]–[6] embed a network into a latent, low-dimensional space that preserves structure proximity and attribute affinity. The resulting compact, low-dimensional vector representations can be then taken as features to any vector-based machine learning algorithms, such as node classification [7], [8], link prediction [9], [10], visualization [11], etc.

However, a notable deficiency in these work is that each node has only one vector representation. But in some applications, one vector cannot fully express the network information. As shown in Fig. 1, (a) is composed of two complete graphs connected by node 4, while (b) has two complete subgraphs sharing node 4. If the traditional network representation learning method is used, that is, only one vector is learned for each node, it is difficult to distinguish these two structures, because in the embedding space, the vectors representation of node 4 are located in the middle position for the two networks. However, node 4 in Fig. 1(b) should exhibit two different attributes, that is, the representation vector of node 4 is located in both the above sub-graph and the below sub-graph, so that it is insufficient to use only one vector to represent it.

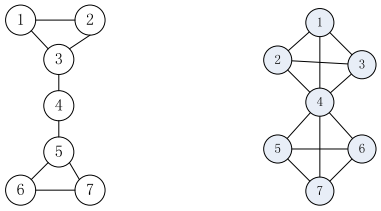


FIGURE 1. Two networks with different structure.

In fact, this problem mentioned above is especially obvious in multi-label networks. Because the label information is implicit in the structure information. Let's take the

BlogCatalog as an example. In this dataset, the nodes represent bloggers. If two bloggers are friends, there is an edge between these two nodes. The labels of each node imply blogger's different hobbies. A node can have multiple labels. If a blogger's hobbies are more extensive, then he will be exposed to different groups of people and will have many friends with different interests, that is, the node denotes this blogger with many labels. Therefore, we need to conduct multi-vector representation research on multi-label networks to reflect the multi-label properties of a node. It should be noted that in the process of training our model, the actual labels are not used, but the structure information of the network is used to learn the multi-label attributes of the node, so our model can also be applied to the general network.

In this paper, based on Skip-gram model [12], we propose a novel multiple vectors embedding approach for multi-label networks(MEMN model). Different from the Skip-gram model only dealing with single vector embedding, our MEMN model can learn multiple vectors for each node. So that each label of a node has its own embedding. Specifically, in our MEMN, each node in the network can have several label vectors and one global vector for further investigation. Meanwhile, maintaining a neighbor cluster center for each label of the node and induce the label by clustering the embeddings of the neighbor nodes (generated by Nodo2vecWalk [6]). The label of a node is predicted as the cluster that is closest to its neighbor vector. The neighbor vector is defined as the average of the global vector representation of the nodes in the neighbor. After predicting the label of a node, performing a gradient update on the embedding of that label. Finally, in order to make best use of these label vectors, setting up a weight for them to form a new vector representation for each node.

Based on the above ideas, here present two methods for our model, MEMN and its non-parametric version NP-MEMN. For MEMN, we set a fixed number K for each node, representing the number of learned label vectors for the node. NP-MEMN dynamically decides the number of label vectors using a hyperparameter λ.

The crucial difference from previous approaches is that each label has its own vector. The label vectors are learned by predicting the label of the node using the current parameter estimates. Overall our paper makes the following contributions:

- We propose an efficient algorithm for feature learning in multi-label networks. we use the emerging label vectors to more accurately perform the clustering.
- A non-parametric variant of our method automatically discovers a varying number of label vectors per node.
- We evaluate MEMN and NP-MEMN for multi-label classification and link prediction on several public datasets.

II. RELATED WORK

Network embedding, as a promising way of network representation, is capable of supporting subsequent network processing and analysis tasks such as node classification, link prediction, network visualization, etc. Modern network embedding methods combines network structure and attributes together to form a more expressive vector. In general, network embedding approaches can be categorized into the following categories:

A. STRUCTURE AND PROPERTY PRESERVING NETWORK EMBEDDING

Among all the information encoded in a network, network structures and properties are two crucial factors that largely affect network inference. DeepWalk [4] and Node2vec [6] demonstrates that random walk based methods can be used to capture the diversity of connectivity patterns in a network. These methods mainly preserve the first order network structure for embedding. LINE [5] is proposed for large-scale network embedding and can preserve the first and second order proximity. Besides these methods for preserving network structure, there are many property preserving network embeddings. Preserving the asymmetric transitivity property of directed network is considered by HOPE [13]. In order to measure the high-order proximity, HOPE summarizes four measurements in a general formulation, that is, Katz Index [14], Rooted PageRank [15], Common Neighbors [15], and Adamic-Adar [16]. SiNE [17] is proposed for signed network embedding, which considers both positive and negative edges in a network. The methods maintain network properties in network embedding space, especially the properties that largely affect the evolution and formation of networks.

B. NETWORK EMBEDDING WITH SIDE INFORMATION

Besides network structures, side information is also valuable. Yang *et al.* [18] propose TADW that takes the rich text information generated by the node associated with nodes into account when they learn the low dimensional representations of nodes. MMDW [19] is a semi-supervised network embedding algorithm and based on the DeepWalk-derived matrix factorization. It adopts support vector machines (SVM) [20] and incorporates the label information to find an optimal classifying boundary. LANE [21] is also proposed to incorporate the label information into the attributed network embedding. Unlike the previous network embedding methods, LANE is mainly based on spectral techniques [22].

C. ADVANCED INFORMATION PRESERVING NETWORK EMBEDDING

Different from side information, the advanced information refers to the supervised or pseudo supervised information in a specific task. Recently, Bourigault *et al.* [23] propose a social network embedding algorithm for predicting information diffusion. The basic idea is to map the observed information diffusion process into a heat diffusion process modeled by a diffusion kernel in the continuous space. The goal of the proposed algorithm is to learn the representations of nodes in the latent space such that the diffusion kernel can best explain the cascades in the training set. The combination

of advanced information and network embedding techniques enables representation learning for networks.

All of the above methods are only one vector representation for each node. There is considerably less prior work on learning multiple vector representations for a node. When facing the high dimensionality of multi-label networks, nodes representation will be overlapping in the hidden space, and embedding nodes to one vector will lose information. For example, a news corpus from New York Times may be tagged as religious, political, financial and educational topics at the same time. This results in the text having an embedding that is approximately the average of its different related field. Therefore, we present the MEMN model, which let each node have multiple vectors.

III. MEMN MODEL

Given an information network $G = (V, E, Y)$, where V is a set of vertices, and $|V|$ denotes the number of vertices in network G. $E \in (V \times V)$ is a set of edges connecting the vertices. $Y \in \mathbb{R}^{|V|\times|\mathcal{Y}|}$ is the vertex label matrix with $\mathcal{Y}$ being a set of labels. If the i-th vertex has the k-th label, the element $Y_{ik} = 1$; otherwise, $Y_{ik} = 0$. For each $(v_i, v_j) \in E$, if network G is undirected, we have $(v_j, v_i) \in E$; if G is directed, (v_j, v_i) unnecessarily belongs to E; Each $(v_i, v_j) \in E$ is also associated with a weight w_{ij}, which is equal to 1, if the network is unweighted.

We formulate feature learning in networks as a maximum likelihood optimization problem. Our analysis is general and applies to any (un)directed, (un)weighted, multi-label network. First, extending the Skip-gram architecture to learn multiple embeddings per node. Let each label of the node have its own embedding. Then, inducing the label by clustering the embeddings of its neighbor of every node $n \in V$. The neighbor of a node are generated through Node2vecWalk [6]. The vector representation of the neighbor is the average of its neighbor nodes' vectors. For every node, maintaining embedding clusters of its label vectors, the number of label vectors in each cluster. The label of a node is predicted as the cluster that is closest to its neighbor vector. After predicting the label of a node, performing a gradient update on the label vector. The process of our model as shown in Fig. 2.

In the MEMN model, we set a hyperparameter K, representing the number of vectors for each node. Each node $n \in V$ is associated with a global vector $v_g(n)$. Each label of the node has an embedding (label vector) $v_l(n, k)(k = 1, 2, \ldots, K)$. Besides, setting a neighbor cluster center $\mu(n, k)(k = 1, 2, \ldots, K)$. We generated several neighbor nodes by Node2vecWalk for each node and refer them as *walk*. All node sequences *walk* form a set *walks*. For each *walk* in *walks*, considering each node n_t and let $c_t = \{n_{t-w}, \ldots, n_{t-1}, n_{t+1}, \ldots, n_{t+w}\}$ be the set of observed neighbor nodes among the window size w. The vector representation of the neighbor is defined as the average of the global vector representation of the nodes in the neighbor. Let $v_{neighbor}(c_t) = \frac{1}{2*w}\sum_{c\in c_t} v_g(c)$ be the vector representation of the neighbor c_t. We use the global vectors of the neighbor nodes to predict the label of the current node. And predict l_t, the label of node n_t when observed with neighbor c_t as the neighbor cluster membership of the vector $v_{neighbor}(c_t)$. More formally,

$$l_t = \arg\max_{k=1,2,\ldots,K} sim(\mu(n_t, k), v_{neighbor}(c_t))$$

In this paper, we consider the similarity between $\mu(n_t, k)$ and $v_{neighbor}(c_t)$ by comparing the opposite of cosine similarity. In order to obtain the loss function of our model, we use Noise Contrastive Estimation (NCE) [30], [31], which allows us to learn a model that provably converges to its objective. For each pair of nodes (n_t, c), the probability that the node c is observed in the neighbor of node n_t given the label of the node n_t is,

$$P(D = 1|v_l(n_t, l_t), v_g(c)) = \frac{1}{1 + e^{-v_l(n_t,l_t)^T v_g(c)}}$$

The probability of not observing node c in the neighbor of n_t given the label of the node n_t is,

$$P(D = 0|v_l(n_t, l_t), v_g(c)) = 1 - P(D = 1|v_l(n_t, l_t), v_g(c))$$

where defining an auxiliary random variable D for node classification, such that $D = 1$ for a node drawn from positive sample and $D = 0$ for a sample drawn from noisy neighbor nodes. The noisy neighbor nodes are randomly sampled from

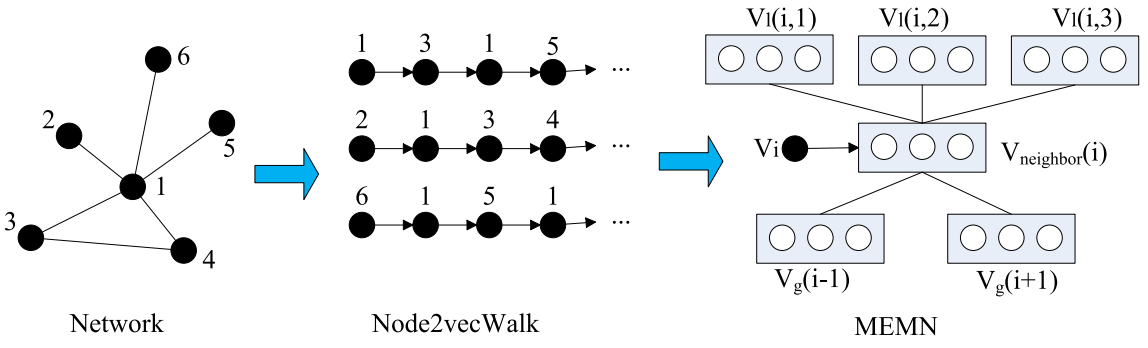


FIGURE 2. The process of MEMN.

the following distribution,

$$P(w) = \frac{p_{unigram}(w)^{\frac{3}{4}}}{Z}$$

where $p_{unigram}(w)$ is the unigram distribution of the nodes and Z is the normalization constant.

A training set *walks* is obtained from Node2vecWalk containing the sequence of node $walk_1, walk_2, \ldots$, the node embeddings are learned by maximizing the objective function:

$$J(\theta) = \sum_{(n_t,c_t)\in D^+} \sum_{c\in c_t} \log P(D=1|v_l(n_t,l_t), v_g(c)) + \sum_{(n_t,c'_t)\in D^-} \sum_{c'\in c'_t} \log P(D=0|v_l(n_t,l_t), v_g(c'))$$

where n_t is the t^{th} node in the sequence $walk_i$, c_t is the set of observed neighbor nodes among the window size w. c'_t is the set of noisy neighbor nodes for the node n_t. D^+ consists of the set of all observed node-neighbor pairs (n_t, c_t) $(t = 1, 2, \ldots, T)$ where T is the length of each *walk* in the sequence. D^- consists of the set of pairs (n_t, c'_t) $(t = 1, 2, \ldots, T)$ where c'_t is the set of randomly sampled, noisy neighbor nodes for the node n_t. Then, we update the neighbor cluster center $\mu(n_t, l_t)$, the number of label vectors $num(n_t, l_t)$ and gradient update on label vectors $v_l(n_t, l_t)$, global vectors $v_g(n_t)$ of nodes in c_t and c'_t.

In order to make best use of each label vector for every node, we use the relation between the number of label vectors $num(n, k)$ in the cluster of the node $n \in V$ for k in $\{1, 2, \ldots, K\}$ and the total number of the vectors in all the neighbor clusters of the node as the weight $P(n, k)$.

IV. NON-PARAMETRIC MEMN MODEL (NP-MEMN)

The MEMN model learns a fixed number of label vectors per node. In this section, the NP-MEMN model which is based on MEMN will be introduced. It is hard to determine the number of node embedding vectors, setting all nodes the same number of embeddings is a huge computational overload, here we provide another way to deal with this issue. First, computing the cosine distance between each existing cluster center $\mu(n, k)$ of node n and its neighbor vector $v_{neighbor}(c_t)$. If the nearest distance is far enough, then creating a new cluster for this node.

We create only one label vector and neighbor cluster for each node on its first occurrence in the training set. When the node is observed with a neighbor are the maximum similarity between the vector representation of the neighbor with every existing cluster center of the node is less than λ, a new neighbor cluster and a label vector are created. Where λ is a hyperparameter of the NP-MEMN model.

Consider the number of neighbor clusters k is the function of the node n_t. When the value of the function $k(n_t)$ is greater than 0, the label of node n_t is given by

$$l_t = \begin{cases} k(n_t)+1, & \text{if } \max_{k=1,2,\ldots,k(n_t)}\{sim \\ & (\mu(n_t,k), v_{neighbor}(c_t))\} < \lambda \\ k_{max}, & \text{otherwise} \end{cases}$$

where $\mu(n_t, k)$ is the cluster center of the k^{th} cluster of node n_t, $k(n_t)$ is the number of clusters of the node n_t and $k_{max} = \arg\max_{k=1,2,\ldots,k(n_t)} sim(\mu(n_t, k), v_{neighbor}(c_t))$. The cluster center is the average of the vector representations of all the neighbor nodes which belong to that cluster. If $l_t = k(n_t) + 1$, a new neighbor cluster and a new label vector are created for the node n_t. The rest of all is the same as MEMN model.

Algorithm 1 Training Algorithm of MEMN Model

Input: network G, walks per vertex r, walk length l, return parameter p, in-out parameter q, vectors per vertex K, window size w, embedding size d.

Initialize $v_l(n,k)$, $v_g(n)$ and $\mu(n,k)$,$\forall n \in V, k \in \{1,\ldots,K\}$ randomly, $num(n,k), \forall n \in V, k \in \{1,\ldots,K\}$ to 0.
$walks$ = Node2vecWalk(G, r, l, p, q)
for *walk* in *walks* **do**
  $T = length(walk)$
  for $t = 1, 2, \ldots, T$ **do**
    $c_t = \{n_{t-w}, \ldots, n_{t-1}, n_{t+1}, \ldots, w_{t+w}\}$
    $v_{neighbor}(c_t) = \frac{1}{2*w}\sum_{c\in c_t} v_g(c)$
    $l_t = \arg\max_{k=1,\ldots,K}\{sim(\mu(n_t,k), v_{neighbor}(c_t))\}$
    Update neighbor cluster center $\mu(n_t, l_t)$ since neighbor c_t is added to neighbor cluster l_t of node n_t.
    Update the count of vectors $num(n_t, l_t)$
    $c'_t = Noisy\_Samples(c_t)$
    Gradient update on $v_l(n_t, l_t)$, global vectors of nodes in c_t and c'_t.
for $i = 1, 2, \ldots, |V|$ **do**
  for $k = 1, 2, \ldots, K$ **do**
    $sum(i) = sum(i) + num(i, k)$
for $i = 1, 2, \ldots, |V|$ **do**
  for $k = 1, 2, \ldots, K$ **do**
    $P(i, k) = num(i, k)/sum(i)$

Output: $v_l(n,k)$,$v_g(n)$, $P(n,k)$, $\forall n \in V, k \in \{1,\ldots,K\}$

V. EXPERIMENT

In this section, first providing an overview of the datasets and baseline methods which will be used in our experiments. Then, evaluating the feature representations obtained through MEMN and NP-MEMN on standard supervised learning tasks: multi-label classification for nodes and link prediction for edges. But our experiments are organized differently than traditional ones. We organize according to the use of multiple vectors: firstly, the experimental effect of MEMN is presented, and then the relationship between the number of labels and the number of vectors is discussed in combination with

the datasets. Finally, the experimental results of NP-MEMN are analyzed by combining the labels.

A. DATASETS

An overview of the datasets we consider in our experiments is given as following:

- BlogCatalog [1]: This is a network of social relationships of the bloggers listed on the BlogCatalog website. The labels represent blogger interests inferred through the meta-data provided by the bloggers. The network has 10312 nodes, 333983 edges, and 39 different labels.
- Protein-Protein Interactions (PPI) [2]: We use a subgraph of the PPI network for Homo Sapiens. The subgraph corresponds to the graph induced by nodes for which we could obtain labels from the hallmark gene sets [24] and represent biological states. The network has 3890 nodes, 76584 edges, and 50 different labels.
- Wikipedia [3]: This is a cooccurrence network of words appearing in the first million bytes of the Wikipedia dump. The labels represent the Part-of-Speech(POS) tags inferred using the Stanford POS-Tagger [25]. The network has 4777 nodes, 184812 edges, and 40 different labels.

B. BASELINE METHODS

To validate the performance of our approach we compare it with several existing graph embedding methods:

- DeepWalk [4]: This approach learns an embedding by sampling random walks from each node, applying Word2Vec-based learning on those walks. The sampling strategy in DeepWalk can be seen as a special case of node2vec with $p = 1$ and $q = 1$.
- LINE [5]: This approach is proposed for large-scale network embedding, and can preserve the first and second order proximity. It learns d-dimensional feature representations in two separate phases. In the first phase, it learns $d/2$ dimensions by BFS-style simulations over immediate neighbors of nodes. In the second phase, it learns the next $d/2$ dimensions by sampling nodes strictly at a 2-hop distance from the source nodes. Last, the two halves are normalized and concatenated.
- Node2Vec [6]: This is a hyperparameter supervised approach that extends DeepWalk by adding two parameters, p and q, so as to control DeepWalk's random walk sampling. For our model-MEMN, when $K = 1$, it corresponds to Node2Vec.
- Graph2Gauss [26]: This approach can learn versatile node embeddings on large-scale attributed graphs, it embeds each node as a Gaussian distribution, and capturing uncertainty about the representation.

C. PARAMETER SETTINGS

In line with previous research [4]–[6], we set the embedding dimensionality d to 128. Moreover, using a large training data to fairly compare to DeepWalk, i.e., walk length $l = 80$, number of walks per node $r = 10$, and window size $w = 10$. For Node2VecWalk, the best in-out and return hyperparameters were learned using 10-fold cross-validation on 10% labeled data with a grid search over $p, q \in \{0.25, 0.50, 1, 2, 4\}$. We tune the parameter (K) over $K \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ for MEMN model. In NP-MEMN we do a search for λ at the set of $\{-0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0, 0.1, 0.2\}$. We train our models using AdaGrad stochastic gradient decent [29] with initial learning rate set to 0.025.

D. EVALUATION METRICS

In our experiment, the task of multi-label classification and link prediction will be performed. For multi-label classification, we adopt $micro - F1$ and $macro - F1$ as many other works do [27]. In detail, for a label A, denoting TP(A), FP(A), and FN(A) as the number of true positives, false positives and false negatives in the instances which are predicted as A, respectively. Suppose $\mathcal{C}$ is the overall label set. $Micro - F1$ and $Macro - F1$ are defined as follows [28]:

- $Macro - F1$ is a metric which gives equal weight to each class. It is defined as follows:

$$Macro - F1 = \frac{\sum_{A \in \mathcal{C}} F1(A)}{|\mathcal{C}|}$$

  where $F1(A)$ is the $F1$-measure for the label A.
- $Micro - F1$ is a metric which gives equal weight to each instance. It is defined as follows:

$$Pr = \frac{\sum_{A \in \mathcal{C}} TP(A)}{\sum_{A \in \mathcal{C}} (TP(A) + FP(A))}$$
$$R = \frac{\sum_{A \in \mathcal{C}} TP(A)}{\sum_{A \in \mathcal{C}} (TP(A) + FN(A))}$$
$$Micro - F1 = \frac{2 * Pr * R}{Pr + R}$$

For the link prediction task, we use AUC and average precision (AP) scores to evaluate the performance. Suppose $\Delta_i(j) = 1$ indicates that the node v_i and v_j have a link, otherwise, $\Delta_i(j) = 0$. We have M * N (M is the number of positive samples, namely, $\Delta_i(j) = 1$, the number of N as negative class samples, namely, $\Delta_i(j) = 0$) positive and negative sample pairs, and use *score* to measure the value of sample pairs. Their definitions are listed as follows:

- AUC is the area under the ROC curve. It is defined as follows:

$$AUC = \frac{\sum_{\Delta_i(j)=1} rank_{(v_i, v_j)} - \frac{M \times (M+1)}{2}}{M \times N}$$

  where *rank* to express the order of sample pairs according to their *score*. For example, the largest score sample pairs, $rank = M + N$, followed by $M + N - 1$.

- *AP* is a metric with good discrimination and stability. It is calculated as follows:

$$P(k) = \frac{|\{(v_i, v_j)|index((v_i, v_j)) \leq k, \Delta_i(j) = 1\}|}{k}$$

$$AP = \frac{\sum_{k=1}^{M+N} P(k)}{M+N}$$

where $index((v_i, v_j))$ is the ranked index of the sample pairs (v_i, v_j).

E. THE EFFECT OF MEMN

In this subsection, we evaluate the feature representation obtained through MEMN on standard supervised learning tasks: multi-label classification for nodes and link prediction for edges.

F. MULTI-LABEL CLASSIFICATION

In the multi-label classification setting, every node is assigned one or more labels from a finite set $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$. We use the vector $y = (y_1, y_2, \ldots, y_m)$ to represent the labels for each node in the network, where $y_i = 1$ refers to $\lambda_i \in \mathcal{L}$. During the training phase, we observe a certain fraction of nodes and all their labels. The task is to predict the labels for the remaining nodes. In our experiments, using the multi-label logistic regression classifier as the base learner.

Due to the methods of DeepWalk, Node2Vec and LINE obtain only one vector for each node, using the individual embedding as the features of nodes directly. For Graph2Gauss, if the competing techniques use an L dimensional embedding, Graph2Gauss's embedding is actually only half of this dimensionality so that the overall number of 'parameters' per node (mean vector + variance terms of the diagonal $\sum_i$) matches L. In our model-MEMN, for each node can obtain the global vector $v_g(n)$, K label vectors $v_l(n, k)$ and their weight $P(n, k)$. Then, obtaining the weighted vector $avg_l(n)$ for each node. More formally,

$$avg_l(n) = \sum_{k=1}^{k=K} P(n, k) * v_l(n, k)$$

Finally, we use the global vector $v_g(n)$ and the weighted vector $avg_l(n)$ as node features to do the multi-label classification task, respectively.

Besides, our methods involve a number of parameters, in order to focus on the effects of multiple vectors, first analyze the effect of hyperparameter K on experimental results. In the experiment, we choose the PPI dataset for analysis, and evaluate the individual embedding $avg_l(n)$ 10 times in order to reduce the noise introduced by the classifier. Then, reporting the average value among the runs. Here, we evaluate accuracy by the Micro-F1 and Macro-F1 scores.

In FIGURE 3, examining how the different choices of K affect the performance of MEMN on the PPI dataset using a 50-50 split between labeled and unlabeled data. Except for the parameter being tested, all other parameters assume default values.

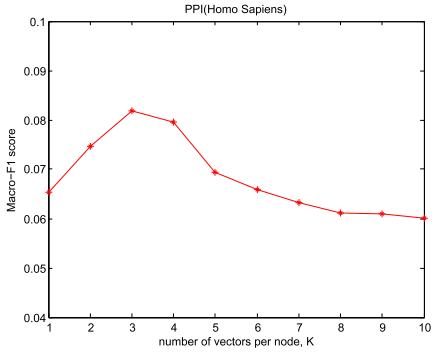


FIGURE 3. Parameter K sensitivity of MEMN on the PPI.

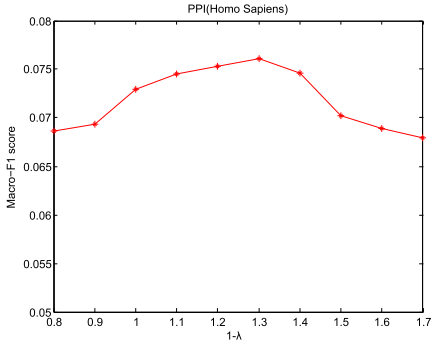


FIGURE 4. Parameter λ sensitivity of NP-MEMN on PPI.

We measure the Macro-F1 score as a function of parameter K. The trends are similar for Micro-F1 and are not shown. From the results in FIGURE 3, we can see with the increase of K, the Macro-F1 score increases first and then decrease. This is due to most nodes have 1 to 3 labels, and only a small number of nodes have a majority of labels. If K is increased again, the performance of embedding will be reduced. So we set $K = 3$ for MEMN on PPI dataset. Similarly, setting $K = 3$ for BlogCatalog and Wikipedia datasets.

Next, we evaluate the experimental results on the multi-label classification task. Similarly, running each experiment 10 times and reporting the average value among the runs. For a more fine-grained analysis, we conduct multiple experiments for each dataset, selecting a random sample of nodes for training and leaving the remaining nodes for testing. Specifically, randomly sampling a portion of the labeled nodes and use them as the training set. In our experiment, increasing the training ratio from 10% to 90%, The results for three datasets,shown in FIGURE 5-6, where MEMN-1 indicates the result of $avg_l(n)$, and MEMN(G) indicates the results of $v_g(n)$. **(In order to save space, the experimental results of MEMN-2 and NP-MEMN will be shown in FIGURE 5 and FIGURE6, and then discuss them separately in the next two subsections.)**

G. LINK PREDICTION

Another important application of network representation learning is link prediction [10], [15], which aims to infer the

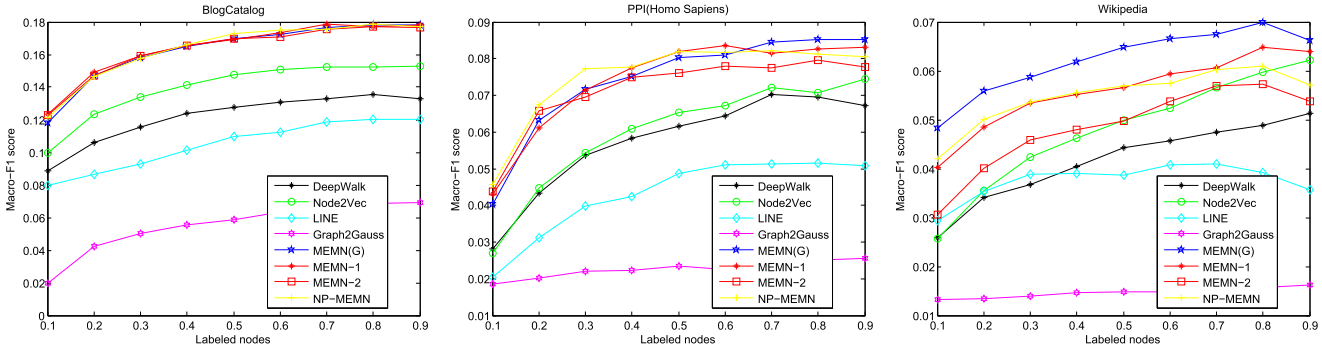


FIGURE 5. Macro-F1 score on varying the amount of labeled data used for training. (a) BlogCatalog dataset. (b) PPI dataset. (c) Wikipedia dataset.

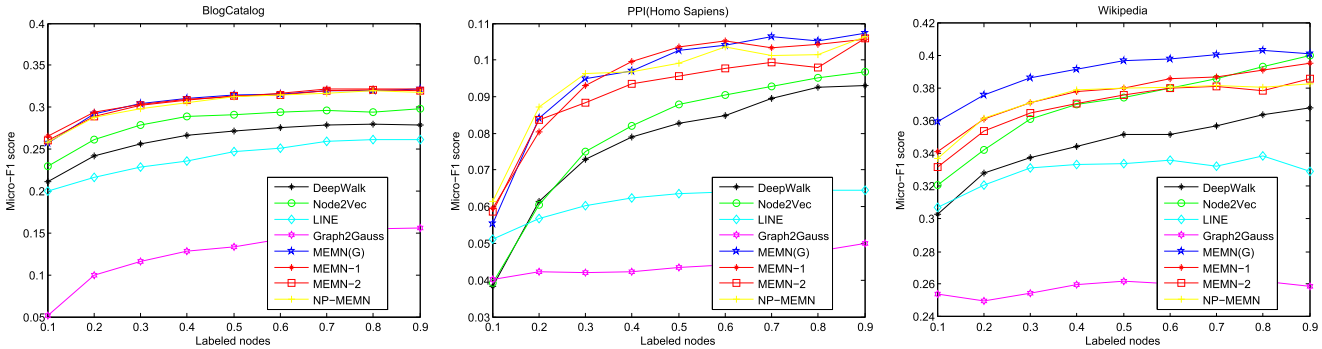


FIGURE 6. Micro-F1 score on varying the amount of labeled data used for training. (a) BlogCatalog dataset. (b) PPI dataset. (c) Wikipedia dataset.

existence of new relationships or still unknown interactions between pairs of nodes based on the currently observed links and their properties. And regarding the node embeddings as their properties.

To evaluate the performance we create a test set that contains 50% randomly selected edges while ensuring that the test set in the graph is connected and equal number of randomly selected non-edges. Then train a classifier to predict whether there is a link between the pairs of nodes or not. As by convention, we report the area under the ROC curve (AUC) and the average precision (AP) scores for each method. To rank the candidate edges, using the dot product of the node embedding as the node pairs' feature. **(similarly, the experimental results of MEMN-2 and NP-MEMN will be shown in TABLE 1.)**

H. EXPERIMENTAL RESULTS

From the results, it is evident that our methods are superior to other state-of-the-art methods. Graph2Gauss showed worse performance than other methods because the method is designed specifically for a graph with attributes. For a more fine-grained analysis, MEMN(G) model consistently outperforms LINE, DeepWalk, Node2Vec and Graph2Gauss on the three datasets. We can also see MEMN(G) performs close to MEMN-1 in the task of multi-label classification on BlogCatalog and PPI. For the Wikipedia dataset, the performance of MEMN(G) is better than MEMN. Whether MEMN-1 or MEMN(G) is a weighting of these K vectors for each node, where the vector $avg_l(n)$ is artificially weighted according to $P(n,k)$, while global vector $v_g(n)$ is weighted by our model mechanisms. We can see the effect of global vector $v_g(n)$ is better than or close to weighted average vector $avg_l(n)$. This phenomenon also has the same embodiment in the link prediction task.

TABLE 1. Link prediction performance.

Algorithm	Dataset					
	BlogCatalog		PPI		Wikipedia	
	AUC	AP	AUC	AP	AUC	AP
DeepWalk	75.22	77.91	87.67	85.64	70.65	77.02
Node2Vec	77.22	79.02	87.80	85.90	74.80	80.29
LINE	71.23	73.45	83.21	82.35	71.46	78.05
Graph2Gauss	54.90	51.73	53.41	52.88	60.44	58.53
MEMN-1	**83.78**	**84.54**	**87.31**	**87.02**	76.09	80.44
MEMN-2	82.34	83.47	86.76	86.09	75.57	80.23
MEMN(G)	83.62	83.65	86.85	86.96	**79.68**	**81.65**
NP-MEMN	83.56	83.76	86.95	86.12	78.53	80.40

I. THE ANALYSIS OF THE RELATIONSHIP

In the previous subsection, we performed a weighted average of the K vectors for each node, but in fact the weights of the

K vectors are not consistent. Therefore, considering another way to use these K vectors: If one of the K vectors of a node has a weight $P(n, k)$ greater than 80%, ignoring the remaining $K-1$ vectors and retain the vector with the largest weight. Otherwise, taking the weighted average of these K vectors as the embedding of this node. More formally,

$$mat_l(n) = \begin{cases} v_{max}(n,k), & \text{if } P_{max}(n,k) > 80\% \\ \sum_{k=1}^{k=K} P(n,k) * v_l(n,k), & \text{otherwise} \end{cases}$$

where $P_{max}(n, k)$ indicates the largest weight of the K vectors, and $v_{max}(n, k)$ indicates the corresponding vector. Finally, using the match vector $mat_l(n)$ as node features to do the multi-label classification and link prediction task, respectively. As is shown in MEMN-2 in FIGURE 5-6 and TABLE 1.

The results show that MEMN-1 and MEMN-2 have similar effects in multi-label classification and link prediction tasks. Next, we combine the dataset for further analysis.

For each dataset, counting the number of labels per node and display it in a histogram, as shown in FIGURE 8. It can be observed that the proportion of nodes with the same number of labels in different datasets is different. In the BlogCatalog dataset, there are 7460 nodes have one label, and the remaining have two or more labels. The number of nodes are respectively accounted for 72.34%, 27.66%. There are 2306 nodes have one label, 1584 nodes have two or more labels, and respectively accounted for 59.28%, 40.72% in the PPI dataset. In Wikipedia dataset, the one label nodes and the remaining are respectively accounted for 68.56%, 31.44%.

For a node with only one label in the datasets, it should theoretically be represented by only one vector. Now we count the weight distribution in the K vectors of each node learned by our model MEMN for this node with only one label. As is shown in FIGURE 7.

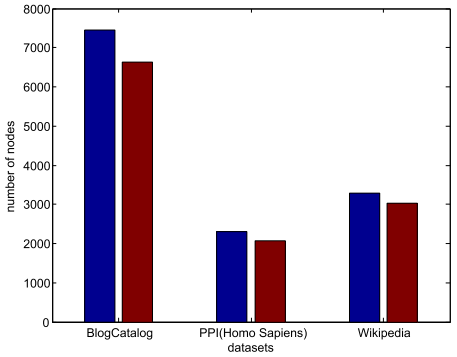


FIGURE 7. The result of MEMN for one label nodes.

In the histogram, the blue part represents the number of nodes with only one label in the datasets, and the red part represents only one label and one of the K vectors learned by the model MEMN has a weight greater than 80%. It can be seen from FIGURE 7 that the number of nodes represented by the blue part are: 7460, 2306, 3275, and the number of nodes represented by the red part are: 6620, 2074, 3022, and the proportion of each part in the blue part is: 88.74%, 89.94%, 92.27%. Since in the K vectors for each node in this subsection, if one of the weights is greater than 80%, we ignore the remaining $K-1$, leaving only the one with the largest weight. Otherwise, taking the weighted average of these K vectors. It can be further verified by the results of FIGURE 7 that almost nodes with only one label have one vector. This further proves that our screening is correct.

J. THE EFFECT OF NP-MEMN

In this subsection, we analyze the effect of NP-MEMN model on multi-label classification and link prediction tasks. First, we still use the PPI dataset to analyze the impact of the choice of hyperparameter λ on the experimental results. Except for the parameter being tested, all other parameters assume default values.

In FIGURE 4, it is shown the result of how the different choices of λ affect the performance NP-MEMN. For the convenience of display the results, using $1-\lambda$ to measure. The experimental results show that the value of it increases first and then decreases, which can be explained in this way: the smaller $1-\lambda$, the more label vectors we learn from our NP-MEMN model. On the contrary, the less. When $1-\lambda = 1.3$, each node almost have 1 to 3 label vectors, which is consistent with the number of labels per node in the real datasets, and there are the best experimental results. So we select $\lambda = -0.3$ for PPI dataset. Similarly, setting $\lambda = -0.5$ for BlogCatalog dataset and $\lambda = -0.3$ for Wikipedia dataset.

Since the NP-MEMN model can learn a different number of vector representations for each node, we process these vectors in a weighted average manner for each node. More formally,

$$npavg_l(n) = \sum_{k=1}^{num(n)} P(n,k) * v_l(n,k)$$

where $num(n)$ indicates the number of learned vectors for node n by NP-MEMN. Then, using the weighted vector $npavg_l(n)$ as the feature representation for each node to do the tasks of multi-label classification and link prediction. The performance of multi-label classification is shown in NP-MEMN in FIGURE 5-6. The performance of link prediction is shown in TABLE 1.

Since in the 5.2 subsection, if one of the K vectors for each node has a weight greater than 80%, only retaining the largest weight, ignoring the remaining $K-1$. Now we use the vectors learned by the NP-MEMN model to count the number of nodes with only one label and one vector, to further verify whether a node of one label has only learned one vector. The result is shown in TABLE 2.

EXPERIMENTAL RESULTS

Through the two tasks of multi-label classification and link prediction, it can be found that the NP-MEMN model and

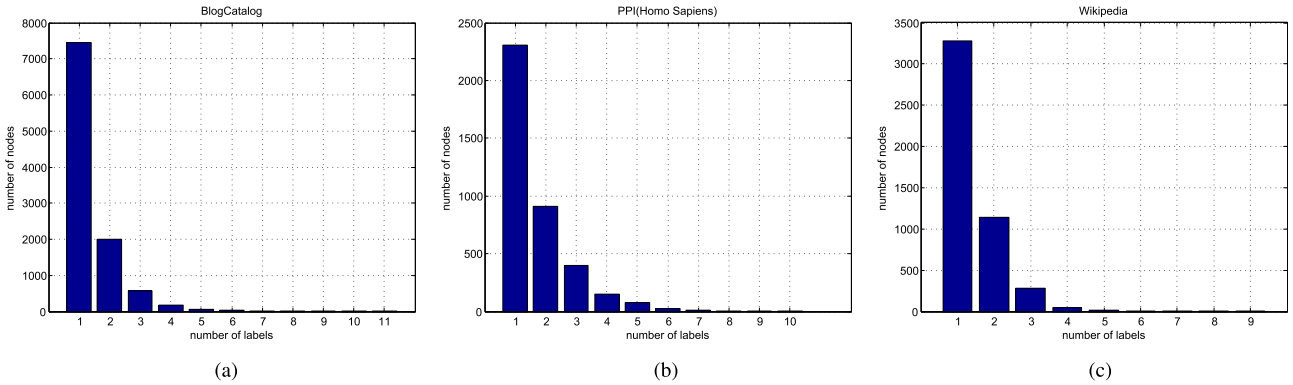


FIGURE 8. The distribution of labels on different datasets. (a) BlogCatalog dataset. (b) PPI dataset. (c) Wikipedia dataset.

TABLE 2. The result of NP-MEMN for one label and one vector.

Datasets	One label	One label one vector	percentage
BlogCatalog	7460	6988	93.67%
PPI	2306	2187	94.84%
Wikipedia	3275	3201	97.74%

the MEMN model have similar experimental effects. And by counting the number of node vectors learned by the NP-MEMN model, it is found that most of the nodes with one label have one vector, which further illustrates the correctness of the practice of retaining only one vector with weights greater than 80% in the MEMN model.

VI. CONCLUSION

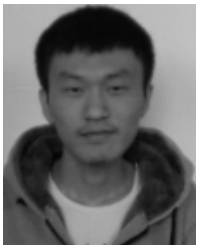
In this paper, we present two methods for multi-label network, MEMN and NP-MEMN. These two methods break the convention that each node has only one vector. On the contrary, our methods can let each label of a node have its own embedding, and induce the label by clustering the embeddings of the neighbor nodes around each node. We fix the number of label vectors to be (K) for MEMN model unless otherwise specified. In NP-MEMN model, we set a hyperparameter λ, which learns varying number of label vectors per node. To demonstrate the efficacy of our model, we design several experiments based on multi-label classification and link prediction in several real-world networks from diverse domains. And analyze the relationship of the number of labels and the number of label vectors. The experimental results show that our models perform better than the baseline methods we set. As a future work, we would like to explore the dynamic network and extend our method for node embedding induction.

REFERENCES

[1] *Social Computing Data Repository*. [Online]. Available: http://socialcomputing.asu.edu/datasets/Twitter
[2] B.-J. Breitkreutz *et al.*, "The BioGRID interaction database," *Nucleic Acids Res.*, vol. 36, pp. D637–D640, 2008.
[3] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[4] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.
[5] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. Int. World Wide Web Conf. Steering Committee*, vol. 2, 2015, pp. 1067–1077.
[6] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.
[7] Z. Shenghuo, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2007, pp. 487–494.
[8] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," *Comput. Sci.*, vol. 16, no. 3, pp. 115–148, 2011.
[9] P. Manisha and R. Kanawati, "Link prediction in complex networks," in *Emerging Automation Techniques for the Future Internet*, pp. 58–97, doi: 10.4018/978-1-4666-9964-9.ch003.
[10] S. Gao, L. Denoyer, and P. Gallinari, "Temporal link prediction by integrating content and structure information," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, 2011, pp. 1169–1174.
[11] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 287–297.
[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[13] M. Ou, P. Cui, J. Pie, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1105–1114.
[14] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.
[15] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proc. 12th Int. Conf. Inf. Knowl. Manage.*, 2003, pp. 556–559.
[16] L. A. Adamic and E. Adar, "Friends and neighbors on the Web," *Social Netw.*, vol. 25, no. 3, pp. 211–230, 2003.
[17] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *Proc. SIAM Int. Conf. Data Mining*, 2017, pp. 327–335.
[18] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. 24th Int. Conf. Artif. Intell.*, 2015, pp. 2111–2117.
[19] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: Discriminative learning of network representation," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 3889–3895.
[20] M. A. Hearst, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul./Aug. 1998.
[21] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 731–739.
[22] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI, USA: American Mathematical Society, 1997.
[23] S. Bourigault, C. Lagnier, S. Lamprier, L. Denoyer, and P. Gallinari, "Learning social network embeddings for predicting information diffusion," in *Proc. 7th ACM Int. Conf. Web Search Data Mining*, 2014, pp. 393–402.

[24] A. Liberzon, A. Subramanian, R. Pinchback, H. Thorvaldsdóttir, P. Tamayo, and J. P. Mesirov, "Molecular signatures database (MSigDB) 3.0," *Bioinformatics*, vol. 27, no. 12, pp. 1739–1740, 2011.
[25] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proc. NAACL*, 2003, pp. 173–180.
[26] A. Bojchevski and S. Günnemann, "Deep Gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking," *CoRR*, vol. abs/1707.03815, 2017.
[27] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining* , Paris, France, Jun./Jul. 2009, pp. 817–826.
[28] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1225–1234.
[29] J. Duchi, E. Hazan, and Y. Singer. "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.
[30] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical model," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, vol. 9, 2010, pp. 297–304.
[31] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proc. Int. Conf. Mach. Learn. Omnipress*, 2012, pp. 419–426.

JUHUA PU received the B.S. degree in computer science from Beijing Jiaotong University, Beijing, China, in 1999, and the Ph.D. degree in computer science from Beihang University (BUAA), Beijing, in 2005. In 2004, she joined The Hong Kong University of Science and Technology as a Visiting Scholar. In 2011, she joined the Georgia Institute of Technology as a Visiting Scholar. She is currently an Associate Professor with BUAA. Her current research interests include urban computing and social network.



ZHUANG LIU was born in Heze, Shandong, China in 1995. He received the B.S. degree from the School of Computer and Control Engineering, Yantai University, China, in 2017. He is currently pursuing the master's degree with the Laboratory of Engineering, Research Center of the Advanced Computer Application Technology, School of Computer Science, Beijing University of Aeronautics and Astronautics. His main research interests are network representation learning and urban computing.

YUJUN CHEN received the B.S. degree in Electronic Engineering Honor Program from China Agricultural University, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Beihang University, Beijing. His research interests broadly lie in learning representation for networked data and data mining.

XINGWU LIU received the B.S. degree in mathematics from East China Normal University, Shanghai, China, in 1999, the M.S. degree in mathematics from Peking University, Beijing, China, in 2002, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Science, Beijing, in 2005. In 2008, he joined INRIA, France, where he held a post-doctoral position. In 2012, he joined the University of Southern California as a Visiting Professor. He is currently an Associate Editor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, and also with the University of Chinese Academy of Sciences, Beijing. His current research interests include distributed computing theory and social networks.