Multi-Layered Network Embedding

Jundong Li*† Chen Chen*† Hanghang Tong* Huan Liu*

Abstract

Network embedding has gained more attentions in recent years. It has been shown that the learned low-dimensional node vector representations could advance a myriad of graph mining tasks such as node classification, community detection, and link prediction. A vast majority of the existing efforts are overwhelmingly devoted to single-layered networks or homogeneous networks with a single type of nodes and node interactions. However, in many real-world applications, a variety of networks could be abstracted and presented in a multi-layered fashion. Typical multi-layered networks include critical infrastructure systems, collaboration platforms, social recommender systems, to name a few. Despite the widespread use of multi-layered networks, it remains a daunting task to learn vector representations of different types of nodes due to the bewildering combination of both within-layer connections and cross-layer network dependencies. In this paper, we study a novel problem of multi-layered network embedding. In particular, we propose a principled framework - MANE to model both within-layer connections and cross-layer network dependencies simultaneously in a unified optimization framework for embedding representation learning. Experiments on real-world multi-layered networks corroborate the effectiveness of the proposed framework.

1 Introduction

The prevalence of various information systems make networked data ubiquitous to our daily life, examples include infrastructure networks, social media networks, brain networks, to name a few. Recent years have witnessed many attempts to gain insights from these networked data by performing different graph learning tasks such as node classification [4, 21, 27], community detection [15, 45] and link prediction [1, 26]. As the very first step, most of these algorithms need to design hand-crafted features to enable the downstream learning problems. One critical issue of these hand-crafted features is that they are problem dependent, and cannot be easily generalized to other learning tasks. To mitigate this problem, recent studies show that through learning general network embedding representations, many subsequent learning tasks could be greatly enhanced [17, 34, 39]. The basic idea is to learn a low-dimensional node vector representation by leveraging the node proximity manifested in the network topological structure.

The vast majority of existing efforts predominately focus on single-layered or homogeneous networks[1]. However, real-world networks are much more complicated as cross-domain interactions between different networks are widely observed, which naturally form a type of multi-layered networks [12, 16, 35]. Critical infrastructure systems are a typical example of multi-layered networks (left part of Figure 1). In this system, the power stations in the power grid are used to provide electricity to routers in the autonomous system network (AS network) and vehicles in the transportation network; while the AS network, in turn, needs to provide communication mechanisms to keep power grid and transportation network work in order. On the other hand, for some coal-fired or gas-fired power stations, a well-functioning transportation network is required to supply fuel for those power stations. Therefore, the three layers in the system form a triangular dependency network [11]. Another example is the organization-level collaboration platform (right part of Figure 1), where the team network is supported by the social network, connecting its employee pool, which further interacts with the information network, linking to its knowledge base. Furthermore, the social network layer could have an embedded multi-layered structure (e.g., each of its layers represents a different collaboration type among different individuals); and so does the information network. In this case, different layers form a tree-structured dependency network.

The availability and widespread of multi-layered networks in real-world has motivated a surge of research. Recent work shows that cross-layer network dependencies are of vital importance in understanding the whole systems and they have an added value over within-layer connectivities. In addition, a small portion cross-layer

*Computer Science and Engineering, Arizona State University, Tempe, AZ, USA. {jundongl, chen_chen, hanghang.tong, huan.liu}@asu.edu

†Indicates Equal Contribution

[1]networks with only a single type of nodes and node-node interactions

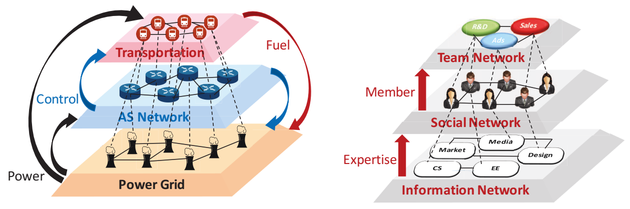


Figure 1: Two typical examples of multi-layered networks: critical infrastructure systems and organization-level collaboration platform.

network dependencies could improve the performance of many learning tasks such as link prediction remarkably [11]. Despite the fundamental importance of studying multi-layered dependencies, the development of a sophisticated learning model which could embed nodes on multi-layered networks into a continuous vector space is still in its infancy. To bridge the gap, in this paper, we study a novel problem on how to perform network embedding for nodes on multi-layered networks. It is a challenging problem mainly because of the following reasons: (1) on multi-layered networks, cross-layer network dependencies are introduced to the system in addition to the within-layer connectivities in each layer. Needless to say, those cross-layer network dependencies play an important role as they also encode the node proximity to some extent. Hence, embedding multi-layered networks would be a non-trivial extension from single layered network embedding, since the latent node features need to capture both within-layer connections and cross-layer network dependencies; (2) nodes on multi-layered networks come from heterogeneous data sources. Even though they are presented in different modalities, they are not mutually independent and could influence each other. Network embedding algorithms should be able to seize their interconnections to learn a unified embedding representation. To tackle the above challenges, we propose a novel multi-layered network embedding framework - MANE. The main contributions are as follows:

- We formally define the problem of multi-layered network embedding;
- We provide a unified optimization framework to model both within-layer connections and cross-layer network dependencies for embedding representation learning on multi-layered networks;
- We provide an effective alternating optimization algorithm for the proposed MANE framework;
- We evaluate the effectiveness of the proposed MANE framework with two real-world multi-layered networks.

Notations	Definitions or Descriptions
$\mathbf{G}$	the layer-layer dependency matrix
$\mathcal{A} = \{\mathbf{A}_1, ..., \mathbf{A}_g\}$	within-layer connectivity matrices
$\mathcal{D} = \{\mathbf{D}_{i,j}, (i \neq j)\}$	cross-layer dependency matrices
g	number of layers
n_i	number of nodes in the i-th layer
$d_1, d_2, ..., d_g$	embedding dimensions
$\mathbf{F}_i$	embedding for the i-th layer

Table 1: Symbols.

2 Problem Definition and Preliminaries

We first summarize the main symbols used in this paper. We use bold uppercase for matrices (e.g., $\mathbf{A}$), bold lowercase for vectors (e.g., $\mathbf{a}$), normal lowercase characters for scalars (e.g., a), and calligraphic for sets (e.g., $\mathcal{A}$). Also, we follow the matrix settings in Matlab to represent the i-th row of matrix $\mathbf{A}$ as $\mathbf{A}(i,:)$, the j-th column as $\mathbf{A}(:,j)$, the (i,j)-th entry as $\mathbf{A}(i,j)$, the transpose of matrix $\mathbf{A}$ as $\mathbf{A}'$, trace of matrix $\mathbf{A}$ as $tr(\mathbf{A})$ if it is a square matrix, Frobenius norm of matrix $\mathbf{A}$ as $\|\mathbf{A}\|_F$. $\mathbf{I}$ denotes the identity matrix.

Next, we introduce the following terminology to ease the understanding of multi-layered networks. Let matrix $\mathbf{G}$ denotes the $g \times g$ layer-layer dependencies in a typical multi-layered network with g layers, where $\mathbf{G}(i,j) = 1$ if the j-th layer depends on the i-th layer, otherwise $\mathbf{G}(i,j) = 0$. Then we use a set of g matrices $\mathcal{A} = \{\mathbf{A}_1, ..., \mathbf{A}_g\}$ to represent the proximity among nodes within each layer. Last, we use a set of matrices $\mathcal{D} = \{\mathbf{D}_{i,j}, (i, j = 1, ..., g)(i \neq j)\}$ to denote the cross-layer network dependencies between different layers. In particular, the matrix $\mathbf{D}_{i,j}$ describes the cross-layer dependencies between the i-th layer and the j-th layer if $\mathbf{G}(i,j) = 1$; otherwise $\mathbf{D}_{i,j}$ is absent. The main symbols are summarized in Table 1. With the above notations, the problem of multi-layered network embedding is defined as follows.

Problem 1. **Multi-Layered Network Embedding**

Given: *the embedding dimension* $d_1, d_2, ..., d_g$ *for different layers; a multi-layered network with: (1) a set of* g *within-layer adjacency matrices* $\mathcal{A} = \{\mathbf{A}_1, ..., \mathbf{A}_g\}$ *where* $\mathbf{A}_i \in \{0,1\}^{n_i \times n_i}$ *(i=1,...,g); and (2) observed cross-layer dependency matrices* $\mathcal{D} = \{\mathbf{D}_{i,j}, (i, j = 1, ..., g)(i \neq j)\}$ *where* $\mathbf{D}_{ij} \in \{0,1\}^{n_i \times n_j}$ *denotes the cross-layer network dependency between* $\mathbf{A}_i$ *and* $\mathbf{A}_j$*;*

Output: *the embedding representation* $\mathbf{F}_i \in \mathbb{R}^{n_i \times d_i}$ *for all nodes in the* i*-th layer* $(i = 1, ..., g)$*.*

3 Proposed Algorithm and Analysis

In this section, we present a novel multi-layered network embedding framework - MANE, which models both within-layer connections and cross-layer network dependencies for embedding representation learning. We first formulate the multi-layered network embedding problem as an optimization problem and then introduce an effective alternating optimization algorithm for the proposed framework. At last, we investigate the time complexity of the MANE framework.

3.1 Within-Layer Connections Modeling The target of a typical embedding algorithm in a single-layered network is to seek for a low-dimensional vector representation which well preserves the node proximity in the original network topological structure. Specifically, for two nodes i and j in a certain layer k, we enforce their embedding vector representations $\mathbf{F}_k(i,:) \in \mathbb{R}^{d_k}$ and $\mathbf{F}_k(j,:) \in \mathbb{R}^{d_k}$ close to each other if these two nodes are interconnected with each other. According to the analysis in spectral theory [29], it can be mathematically formulated by minimizing the following objective function:

$$\min_{\mathbf{F}_k} \frac{1}{2} \sum_{i,j} \mathbf{A}_k(i,j) \| \frac{\mathbf{F}_k(i,:)}{\sqrt{Deg_i}} - \frac{\mathbf{F}_k(j,:)}{\sqrt{Deg_j}} \|_2^2, \quad (3.1)$$

where Deg_i and Deg_j are degrees of nodes i and j respectively. It is easy to verify that the above optimization problem can be reformulated as the following maximization problem:

$$\max_{\mathbf{F}_k} tr(\mathbf{F}_k' \mathbf{L}_k \mathbf{F}_k) \text{ s.t. } \mathbf{F}_k' \mathbf{F}_k = \mathbf{I}, \quad (3.2)$$

where $\mathbf{L}_k = \mathbf{H}_k^{-1/2} \mathbf{A}_k \mathbf{H}_k^{-1/2}$ is the normalized Laplacian matrix for the network structure in the k-th layer, and $\mathbf{H}_k$ is a diagonal matrix with its diagonal element as $\mathbf{H}_k(i,i) = \sum_{j=1}^{n_k} \mathbf{A}_k(i,j)$. It should be mentioned that the constraint $\mathbf{F}_k'\mathbf{F}_k = \mathbf{I}$ is imposed in the above objective function to avoid arbitrary scaling factor of the embedding representation $\mathbf{F}_k$.

Putting the within-layer connections modeling for all g layers together, we obtain the following objective function:

$$\max_{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_g} \sum_{i=1}^{g} tr(\mathbf{F}_i' \mathbf{L}_i \mathbf{F}_i) \text{ s.t. } \mathbf{F}_i' \mathbf{F}_i = \mathbf{I} \ (\forall i = 1,,,.,g). \quad (3.3)$$

The above optimization problem is equivalent to solving a set of g trace maximization problems separately. For each sub-problem in a certain layer k, we could obtain the optimal solution $\mathbf{F}_k$ by concatenating the top d_k eigenvectors of $\mathbf{L}_k$ which correspond to the largest d_k eigenvalues.

3.2 Cross-Layer Network Dependencies Modeling As dependencies in a multi-layered network consist of both within-layer connectivities and cross-layer network dependencies, we now focus on discussing how to model the cross-layer network dependencies for an embedding representation for nodes from different layers. In the previous subsection, we enforce the embedding representations of two nodes to be close to each other in the Euclidean space if they are connected in the same layer. However, it is not easy to measure the proximity of nodes from two different layers directly as they come from different domains, which hinders their comparability. Usually, the embedding representation of nodes can be interpreted as the latent features, and these latent features interact with each other to generate cross-layer network dependencies. For example, a conventional recommender systems can be regarded as a two layer networks, with user-item ratings as the cross-layer network dependencies. In such networks, users could be characterized by some latent features, representing her/his personal interests or tastes; and also, each item can be represented by some latent features, showing its key properties or compositions. The latent features of users and items interact with each other to form the user-item ratings. Motivated by the block models [28], we model the node proximity of nodes from different layers by investigating how their latent features' interplay approximates the real dependencies. Specifically, let $\mathbf{K}_{ij} \in \mathbb{R}^{d_i \times d_j}$ indicates the interaction matrix between the embeddings of the i-th layer and the embeddings of the j-th layer, then the node proximity between nodes from layer i and layer j can be measured by solving the following matrix approximation problem:

$$\begin{aligned} &\min_{\mathbf{F}_i, \mathbf{F}_j, \mathbf{K}_{ij}} \|\mathbf{D}_{ij} - \mathbf{F}_i \mathbf{K}_{ij} \mathbf{F}_j'\|_F^2 \\ &\text{s.t. } \mathbf{F}_i'\mathbf{F}_i = \mathbf{I}, \mathbf{F}_j'\mathbf{F}_j = \mathbf{I}. \end{aligned} \quad (3.4)$$

Similarly, the orthogonal constraints in the above formulation are incorporated to avoid arbitrary scaling factors. By incorporating all possible cross-layer network dependencies together, node proximity for all cross-layers can be measured by solving the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{F}_i, \mathbf{F}_j, \mathbf{K}_{ij}} \sum_{i,j=1}^{g} \|\mathbf{D}_{ij} - \mathbf{F}_i \mathbf{K}_{ij} \mathbf{F}_j'\|_F^2 \\ &\text{s.t. } \mathbf{F}_i'\mathbf{F}_i = \mathbf{I} \ (\forall i = 1, ..., g). \end{aligned} \quad (3.5)$$

3.3 Proposed Embedding Framework - MANE We have shown how to model within-layer connections and cross-layer network dependencies for embedding representation learning in a typical multi-layered network in Eq. (3.3) and Eq. (3.5), respectively. By combing these two objective functions together, we obtain the final objective function for multi-layered network embedding framework - MANE:

$$\max_{\mathbf{F}_i,\mathbf{K}_{ij}} \sum_{i=1}^{g} tr(\mathbf{F}_i'\mathbf{L}_i\mathbf{F}_i) - \alpha \sum_{i,j=1}^{g} \|\mathbf{D}_{ij} - \mathbf{F}_i\mathbf{K}_{ij}\mathbf{F}_j'\|_F^2 \quad \text{s.t. } \mathbf{F}_i'\mathbf{F}_i = \mathbf{I}\ (\forall i = 1,...,g), \tag{3.6}$$

where the positive parameter α is introduced to balance the contributions of within-layer connectivities modeling and cross-layer network dependencies modeling for embedding learning. Further analysis on the effect of the positive parameter α will be investigated later.

The objective function in Eq. (3.6) has two sets of variables, the embedding representation $\mathbf{F}_i$ for each layer and the layer-layer embedding interaction matrices $\mathbf{K}_{ij}$. It is easy to verify that that the objective function is not convex w.r.t. to these two sets of variables. Meanwhile, due to the orthogonal constraints, it is not easy to obtain a global optimal solution. Next, we present an effective iterative algorithm to solve the optimization problem in Eq. (3.6).

3.3.1 Computation of $\mathbf{K}_{ij}$ To update matrix $\mathbf{K}_{ij}$, we remove the terms that are irrelevant to $\mathbf{K}_{ij}$, then Eq. (3.6) can be reformulated as:

$$\min_{\mathbf{K}_{ij}} \|\mathbf{D}_{ij} - \mathbf{F}_i\mathbf{K}_{ij}\mathbf{F}_j'\|_F^2. \tag{3.7}$$

We have the following theorem for the solution of $\mathbf{K}_{ij}$.

THEOREM 3.1. *If $\mathbf{F}_i$ and $\mathbf{F}_j$ are fixed, the optimal solution of $\mathbf{K}_{ij}$ in Eq. (3.7) can be obtained as:*

$$\mathbf{K}_{ij} = \mathbf{F}_i'\mathbf{D}_{ij}\mathbf{F}_j. \tag{3.8}$$

Proof. We expand Eq. (3.7) as follows:

$$\begin{aligned} &\|\mathbf{D}_{ij} - \mathbf{F}_i\mathbf{K}_{ij}\mathbf{F}_j'\|_F^2 \\ &= tr[(\mathbf{D}_{ij} - \mathbf{F}_i\mathbf{K}_{ij}\mathbf{F}_j')(\mathbf{D}_{ij} - \mathbf{F}_i\mathbf{K}_{ij}\mathbf{F}_j')'] \\ &= tr(\mathbf{D}_{ij}\mathbf{D}_{ij}') + tr(\mathbf{F}_i\mathbf{K}_{ij}\mathbf{K}_{ij}'\mathbf{F}_i') - 2tr(\mathbf{F}_i\mathbf{K}_{ij}\mathbf{F}_j'\mathbf{D}_{ij}') \\ &= tr(\mathbf{D}_{ij}\mathbf{D}_{ij}') + tr(\mathbf{F}_i'\mathbf{F}_i\mathbf{K}_{ij}\mathbf{K}_{ij}') - 2tr(\mathbf{F}_i\mathbf{K}_{ij}\mathbf{F}_j'\mathbf{D}_{ij}') \\ &= tr(\mathbf{D}_{ij}\mathbf{D}_{ij}') + tr(\mathbf{K}_{ij}\mathbf{K}_{ij}') - 2tr(\mathbf{F}_i\mathbf{K}_{ij}\mathbf{F}_j'\mathbf{D}_{ij}'). \end{aligned} \tag{3.9}$$

By setting the derivative of the above formulation w.r.t. $\mathbf{K}_{ij}$ to zero, we obtain the closed-form solution of $\mathbf{K}_{ij}$:

$$\mathbf{K}_{ij} = \mathbf{F}_i'\mathbf{D}_{ij}\mathbf{F}_j, \tag{3.10}$$

which completes the proof.

3.3.2 Computation of $\mathbf{F}_i$ We have shown how to obtain the optimal solution of $\mathbf{F}_i$ when all the other variables are fixed. We have the following theorem:

THEOREM 3.2. *The maximization problem in Eq. (3.6) is equivalent to the following optimization problem:*

$$\max_{\mathbf{F}_i} \sum_{i=1}^{g} tr(\mathbf{F}_i'\mathbf{L}_i\mathbf{F}_i) + \alpha \sum_{i,j=1}^{g} tr(\mathbf{F}_i'\mathbf{D}_{ij}\mathbf{F}_j\mathbf{F}_j'\mathbf{D}_{ij}'\mathbf{F}_i) \quad s.t.\ \mathbf{F}_i'\mathbf{F}_i = \mathbf{I},\ (\forall i = 1,...,g) \tag{3.11}$$

Proof. By plugging the solution of $\mathbf{K}_{ij}$ in Eq. (3.8) into Eq. (3.6), the second term (excluding α) in Eq. (3.6) can be reformulated as:

$$\begin{aligned} &-\sum_{i,j=1}^{g} \|\mathbf{D}_{ij} - \mathbf{F}_i\mathbf{K}_{ij}\mathbf{F}_j'\|_F^2 \\ &= -\sum_{i,j=1}^{g} tr(\mathbf{D}_{ij}\mathbf{D}_{ij}') - \sum_{i,j=1}^{g} tr(\mathbf{F}_i'\mathbf{D}_{ij}\mathbf{F}_j\mathbf{F}_j'\mathbf{D}_{ij}'\mathbf{F}_i) \\ &\quad + 2\sum_{i,j=1}^{g} tr(\mathbf{F}_i\mathbf{F}_i'\mathbf{D}_{ij}\mathbf{F}_j\mathbf{F}_j'\mathbf{D}_{ij}') \\ &= -\sum_{i,j=1}^{g} tr(\mathbf{D}_{ij}\mathbf{D}_{ij}') + \sum_{i,j=1}^{g} tr(\mathbf{F}_i'\mathbf{D}_{ij}\mathbf{F}_j\mathbf{F}_j'\mathbf{D}_{ij}'\mathbf{F}_i). \end{aligned} \tag{3.12}$$

As $\sum_{i,j=1}^{g} tr(\mathbf{D}_{ij}\mathbf{D}_{ij}')$ is constant, by combining Eq. (3.12) with the first term of Eq. (3.6), we complete the proof.

To update $\mathbf{F}_i$, we remove the terms that are irrelevant to $\mathbf{F}_i$, then the optimization problem can be reformulated as:

$$\max_{\mathbf{F}_i} tr(\mathbf{F}_i'\mathbf{L}_i\mathbf{F}_i) + \alpha \sum_{j=1}^{g} tr(\mathbf{F}_i'\mathbf{D}_{ij}\mathbf{F}_j\mathbf{F}_j'\mathbf{D}_{ij}'\mathbf{F}_i) \quad \text{s.t. } \mathbf{F}_i'\mathbf{F}_i = \mathbf{I}. \tag{3.13}$$

It is easy to verify that $\mathbf{M}_i = \mathbf{L}_i + \alpha \sum_{j=1}^{g} \mathbf{D}_{ij}\mathbf{F}_j\mathbf{F}_j'\mathbf{D}_{ij}'$ is a positive-semidefinite matrix. Therefore, the optimization problem w.r.t. $\mathbf{F}_i$ boils down to a trace maximization problem with orthogonality constraint, which has a closed-form solution according to the Ky-Fan theorem [5]. Specifically, $\mathbf{F}_i$ could be obtained by concatenating the top d_i eigenvectors of $\mathbf{M}_i$ which correspond to the largest d_i eigenvalues.

In summary, the objective function of the proposed multi-layered network embedding framework - MANE in Eq. (3.6) is solved in an iterative way as illustrated in Algorithm 1.

3.4 Time Complexity Analysis First, we denote the number of edges in each within-layer connectivity matrix $\mathbf{A}_i$ as m_i. Meanwhile, the number of edges in

Algorithm 1 Multi-layered network embedding - MANE

```
Input: A multi-layered network with layer-layer dependency matrix $\mathbf{G}$; a set of within-layer connectivity matrices $\mathcal{A} = \{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_g\}$; a set of cross-layer dependency matrices $\mathcal{D} = \{\mathbf{D}_{i,j}, (i, j = 1, ..., g)\}$; embedding dimension $d_1, ..., d_g$; model parameter $\alpha$
Output: Embedding representation $\mathcal{F} = \{\mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_g\}$.
1: Initialize $\mathbf{F}_i$ $(1 \leq i \leq g)$ with orthonormal matrices;
2: Calculate $\mathbf{L}_i$ $(1 \leq i \leq g)$;
3: while objective function value in Eq. (3.6) not converge do
4:     Calculate $\mathbf{M}_i$ $(1 \leq i \leq g)$;
5:     Obtain $\mathbf{F}_i$ $(1 \leq i \leq g)$ by concatenating the top $d_i$ eigenvectors of $\mathbf{M}_i$;
6: end while
```

each cross-layer dependency matrix $\mathbf{D}_{ij}$ is m_{ij}. As both the within-layer connectivity matrix and cross-layer dependency matrix are very sparse, we have $m_i \ll n_i^2$ and $m_{ij} \ll n_i n_j$. Before the iteration process in Algorithm 1, the complexity of computing the Laplacian matrix $\mathbf{L}_i$ is $\mathcal{O}(m_i)$. The complexity of computing $\mathbf{M}_i$ is $\mathcal{O}(\sum_{i=1}^{g}(n_i^2 d_j + m_{ij} d_j))$. In addition to that, the computation of obtaining the top d_i eigenvectors of matrix $\mathbf{M}_i$ is $\mathcal{O}(m_i d_i + n_i d_i^2)$ if we adopt the Lanczos method. Since $\mathbf{M}_i$ is not a sparse matrix, the complexity is equivalent to $\mathcal{O}(n_i^2 d_i)$. Based on the above analysis, the time complexity of the proposed MANE framework is $\#iter \times \mathcal{O}(\sum_{i=1}^{g}(m_i n_i + \sum_{i=1}^{g}(n_i^2 d_j + m_{ij} d_j) + n_i^2 d_i))$.

4 Evaluations

In this section, we conduct experiments to evaluate the effectiveness of MANE on the multi-class node classification task. We first introduce the datasets, baseline methods, and experimental settings before presenting the details of experiments. Last, we perform a parameter sensitivity study to show how the model parameters affect the performance of MANE.

4.1 Dataset We use two datasets to demonstrate the effectiveness of MANE, including one academic collaboration network (Aminer) and one infrastructure system network (Infra).

Aminer. The Aminer dataset is an academic collaboration platform in the domain of computer science[2] which is naturally a three-layered network. The entities in these three layers are papers, authors and conference venues. Specifically, each layer of the network represents (1) a co-authorship network, (2) a paper-paper citation network, and (3) a venue-venue citation network respectively. In addition, two types of cross-layer network dependencies naturally exist, including "who-writes-which paper", and "which venue-publishes-which paper". The labels of the venues are classified by the research areas in Google Scholar[3]. Overall, we have eight different research areas[4]. The labels of papers and authors are also determined by these eight research areas. Specifically, the label of each paper is decided by the label of the corresponding venue. And we label each author according to the major areas of his/her publications.

Dataset	Aminer	Infra
# of Layers	3	3
# of Nodes	17,504	8,325
# of Links	107,466	15,138
# of CrossLinks	35,229	23,897
# of Labels	8	5

Table 2: Statistics of Datasets.

Infra. This dataset contains three critical infrastructure networks (1) an airport network[5], (2) an AS network[6] and (3) a power grid [42]. The three infrastructure networks are functionally dependent on each other. Therefore, they form a triangle-shaped multi-layered network. The labels of the nodes are determined by service areas inferred from the geographic proximity.

The statistics of the datasets are listed in Table 2.

4.2 Compared Methods The compared methods can be roughly categorized into two classes: matrix factorization based methods and single-layered network embedding methods. Among them, CCF, CMF and NMF are matrix factorization based methods. The rest methods fall into the category of network embedding based approaches.

- **CCF** [46] is a collective collaborative filtering approach that works on the multi-layered networks. The within-layer connections are incorporated by the social homophily effect.
- **CMF** [37] is a collective matrix factorization approach that jointly approximates within-layer connections and cross-layer dependencies on the multi-layered networks.

[2]https://aminer.org/

[3]`https://scholar.google.com/citations?viewop=topvenues&hl=en&vq=eng.`

[4]1. Computer Systems, 2. Computer Networks & Wireless Communication, 3. Computational Lingustics, 4. Computer Graphics, 5. Human Computer Interaction, 6. Theoretical Computer Science, 7. Computer Vision & Pattern Recognition and 8. Database & Information Systems

[5]`http://www.levmuchnik.net/Content/Networks/NetworkData.html.`

[6]`http://snap.stanford.edu/data/`

- **NMF** [24] is a nonnegative matrix factorization method that works on the *flattened* single-layered network of the multi-layered networks. Specifically, the diagonal blocks in the flattened network contains all the within-layered connections; while the off-diagonal blocks contains the corresponding cross-layer dependencies.
- **Deepwalk** [34] is a recent proposed network embedding approach that learns node embedding representation by language model. In the experiments, we perform Deepwalk on the flattened single-layered network of the original multi-layered network. Hence, a walk may contain both within-layer connections and cross-layer dependencies in an unstructured manner.
- **LINE** [39] is a large-scale network embedding method that preserves both first-order and second-order node proximity. Similar to Deepwalk, LINE is also performed on the flattened single-layered network of the original multi-layered networks.
- **Deepwalk-within** is a variant of Deepwalk but only considers the within-layer connections for embedding representation learning.
- **LINE-within** is a variant of LINE. Similar to Deepwalk-within, it only leverages the within-layer node connections for embedding learning.
- **Metapath2vec** [14] is a heterogeneous network embedding method which can effectively leverage the semantic correlations between different types of nodes with metapath guided random walks. For Aminer dataset, we generate random walks by metapaths "APA" and "APVPA"[7]. While on Infra, we exploit the following metapaths "APA", "APNPA", "APNPA", "ANA", and "ANPNA"[8].

4.3 Experimental Settings Following the commonly adopted way [34, 39], we evaluate the proposed framework MANE on the multi-class node classification task. Specifically, after we obtain the embedding representation for nodes in each layer, we split these nodes into training set and testing set. The ratio of training data is varied from 10% to 90%. We choose the logistic regression as the discriminative classifier. Two widely used evaluation criteria based on F1-score, i.e., Macro-F1 and Micro-F1 are used to measure the performance of multi-class classification algorithms [34]. Micro-F1 can be considered as a weighted average of F1-score over all k different class labels. Macro-F1 is an arithmetic average of F1-scores of all output class labels:

$$\text{Micro-F1} = \frac{\sum_{i=1}^{k} 2\text{TP}^i}{\sum_{i=1}^{k}(2\text{TP}^i + \text{FP}^i + \text{FN}^i)} \tag{4.14}$$
$$\text{Macro-F1} = \frac{1}{k}\sum_{i=1}^{k} \frac{2\text{TP}^i}{(2\text{TP}^i + \text{FP}^i + \text{FN}^i)},$$

where TP^i, FP^i, and FN^i denote the number of positives, false positives, and false negatives in the i-th class, respectively. Normally, the higher the values are, the better the classification performance is.

We follow the suggestions of the original papers to specify the parameters for baseline methods. In the experiments, the embedding dimension for all methods and all datasets is set to be 100. We run all the experiments 10 times and report the average performance. More discussions on the parameter sensitivity study of MANE will be investigated later.

4.4 Node Classification Performance We first assess the quality of obtained embedding representation on the task of multi-class node classification. The comparison results w.r.t. Macro-F1 and Micro-F1 are shown in Table 3 and Table 4. We make the following observations:

- When the percentage of training data is varied from 10% to 90%, the classification performance of all methods gradually increase.
- In almost all cases, the proposed MANE framework outperforms all baseline methods by obtaining higher classification performance. We also perform a pairwise Wilcoxon signed rank test between MANE and all other baseline methods, and the test results show that MANE is significantly better, with a significance level of 0.05.
- Cross-layer network dependencies can help boost the classification performance as all methods that work with the cross-layer dependencies (e.g., MANE, Metapath2vec, Deepwalk, LINE, CCF, CMF, NMF) are superior to the methods with only within-layer connections (e.g., Deepwalk-within, and LINE-within).
- Methods that work under the flattened homogeneous networks (e.g., NMF, Deepwalk, LINE, Metapath2vec) are inferior to the methods that explicitly differentiate different types of links (e.g., CCF, CMF and MANE). This observation indicates that it is necessary to discriminate the modeling of within-layer connectivities and cross-layer network dependencies for embedding representation learning.

[7]'A' = Author; 'P' = Paper; 'V' = Venue
[8]'A' = Airport; 'P' = Power Station; 'N' = Network Router

Training Ratio		10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	CCF	85.34	88.37	89.70	90.91	91.07	91.24	91.31	91.44	91.62
	CMF	84.33	87.65	88.84	89.39	90.38	90.43	90.47	90.50	91.20
	NMF	74.40	75.52	75.97	76.49	80.91	84.88	85.20	86.00	86.79
	Deepwalk	82.67	85.05	86.02	86.81	87.32	87.37	87.47	88.01	88.14
	Deepwalk-within	63.92	67.33	68.48	68.87	69.89	70.50	71.70	71.87	72.27
	LINE	83.59	84.85	86.90	87.53	88.09	88.15	88.34	88.37	88.50
	LINE-within	44.72	51.80	56.58	61.58	63.27	66.55	67.69	68.21	70.85
	Metapath2vec	85.46	86.53	87.23	87.71	88.06	89.45	89.42	89.99	90.80
	MANE	**88.80**	**90.46**	**91.15**	**91.78**	**92.31**	**92.37**	**92.38**	**92.43**	**92.72**
Micro-F1	CCF	92.44	92.89	92.87	93.35	93.76	93.94	94.40	94.38	94.48
	CMF	92.07	92.88	92.62	93.10	93.25	93.57	94.18	94.30	92.64
	NMF	88.06	88.28	88.48	88.73	89.42	89.55	89.80	90.07	90.36
	Deepwalk	89.99	90.54	90.82	91.08	91.33	91.59	91.72	91.84	92.03
	Deepwalk-within	83.23	84.21	84.70	84.75	85.08	85.11	85.52	85.69	86.32
	LINE	89.16	90.83	91.40	91.74	91.90	91.93	91.96	92.05	92.20
	LINE-within	66.51	72.27	73.97	75.74	76.61	77.34	78.13	78.45	79.36
	Metapath2vec	92.16	93.51	93.77	93.89	93.74	93.93	93.94	94.37	94.96
	MANE	**93.15**	**94.44**	**94.73**	**94.75**	**95.19**	**95.28**	**95.31**	**95.45**	**95.59**

Table 3: Node classification performance comparison on AMINER dataset with different portions of training data.

Training Ratio		10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	CCF	53.43	54.72	55.13	55.85	56.15	56.42	56.38	56.89	56.92
	CMF	54.63	54.56	55.12	55.77	56.30	56.91	58.01	58.27	58.47
	NMF	46.01	47.42	49.84	50.62	51.26	51.67	51.75	52.35	53.06
	Deepwalk	52.64	54.24	54.64	55.34	55.42	55.87	55.92	56.11	56.36
	Deepwalk-within	47.56	51.37	50.40	51.77	51.95	52.69	52.79	53.03	53.41
	LINE	53.41	53.91	55.03	55.33	55.59	55.87	55.94	55.89	56.15
	LINE-within	47.05	48.93	49.15	50.15	50.96	51.03	51.57	52.11	52.20
	Metapath2vec	54.34	55.39	55.88	57.59	**58.79**	**59.86**	58.58	59.29	59.27
	MANE	**56.68**	**57.32**	**57.54**	**58.42**	58.57	58.50	**59.38**	**59.35**	**59.64**
Micro-F1	CCF	56.00	58.15	58.27	59.40	59.77	59.97	60.88	61.16	61.58
	CMF	56.91	59.57	59.29	60.55	60.93	61.46	62.35	62.61	62.87
	NMF	52.66	54.21	54.68	56.04	56.32	56.54	57.05	57.13	57.41
	Deepwalk	55.59	56.31	57.29	58.20	58.30	58.35	59.01	59.38	59.72
	Deepwalk-within	52.24	56.28	55.04	58.54	58.19	57.20	57.45	57.74	57.90
	LINE	53.74	55.49	57.27	58.08	58.82	59.60	60.17	60.43	60.64
	LINE-within	46.18	46.66	52.28	53.07	53.62	53.99	53.67	54.49	55.44
	Metapath2vec	55.44	56.93	59.28	59.37	59.59	61.50	60.97	61.96	61.33
	MANE	**59.40**	**61.67**	**62.62**	**62.91**	**63.12**	**63.37**	**63.82**	**64.20**	**64.81**

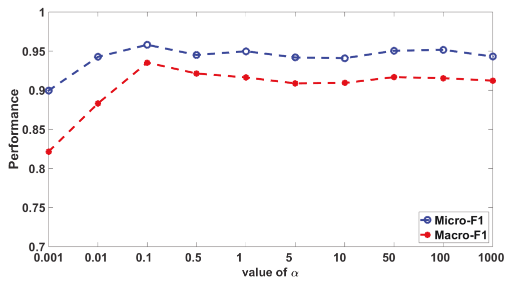
Table 4: Node classification performance comparison on INFRA dataset with different portions of training data.

4.5 Parameter Study The proposed MANE framework has one important parameter α, which is used to balance the contribution of within-layer connectivities modeling and cross-layer network dependencies modeling for embedding representation learning. We first vary the value of α and investigate how it affects the classification performance. Specifically, the training ratio is specified as 80% and the embedding dimension is set to be 100. As can be shown in Figure 2(a), with the increase the value of α, the classification performance first increases, reaches its peak, and then becomes relatively stable. The classification performance is the best when α is around 0.1. Next, to study the impact of the embedding dimension d, we vary the value from 20 to 120. The variation result is shown in Figure 2(b), we can see that by increasing the embedding dimension d, the classification performance first increases and then keeps stable. Hence, we empirically set it as 100 in the experiments. It should be noted that we only perform the parameter sensitivity study on the AMINER dataset as we have similar observations on the INFRA dataset.

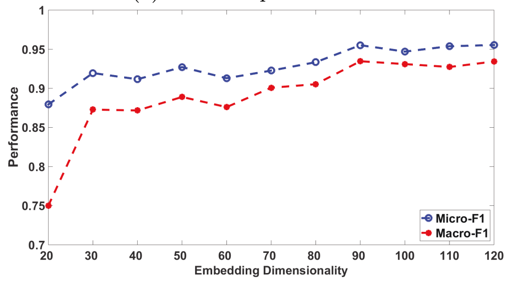
5 Related Work

In this section, we review related work from two aspects: multi-layered networks and network embedding.

5.1 Multi-Layered Networks Multi-layered networks have attracted considerable research attentions in recent years. To construct a general overview of this research area, Kivela et al. [22] provided a comprehensive survey about different types of multi-layered networks, including multi-modal networks [18], multi-dimensional networks [3], multiplex networks [2] and interdependent



(a) Effect of parameter α.



(b) Effect of embedding dimension d.

Figure 2: Parameter study on the AMINER dataset.

networks [6]. The multi-modal networks in [18] is defined as a 3-tuple (V, E, M), in which the edges in E are hyperedges that have multiple source nodes and target nodes from V; and each edge can have different mode from M. On multi-dimensional networks and multiplex networks, different layers of the network represent various types of connections between the same set of nodes. The network studied in our paper falls in the category of interdependent networks. Due to the complex dependency relationships in the network, most of the previous research centers around two-layered networks [6, 16, 33, 36], with a few exceptions that focus on arbitrarily structured multi-layered interdependent networks [9, 10]. Other related studies that infuse networks or data from multiple domains include cross-layer dependency inference [11], cross-network ranking [30] and clustering [31], and multi-view data analysis [48, 49]. To the best of our knowledge, this is the first attempt to study embedding representation learning on multi-layered interdependent networks.

5.2 Network Embedding Network embedding has become an effective tool in analyzing real-world networks. The learned low-dimensional node vector representation not only can help us gain more insights on the essential properties of the underlying networks, but also can benefit many downstream learning tasks, such as node classification, community detection, link prediction and anomaly detection. Perozzi et al. [34] incorporated language modeling techniques in NLP community into social network analysis and learn node embeddings by truncated random walks. Grover and Leskovec [17] extended the Deepwalk method by adding flexibility in exploiting node neighborhoods. Tang et al. [39] considered both first order and second order node proximity into embedding representation learning. Cao et al. [7] further extended LINE to leverage high-order node proximity in network embedding. In [19, 20, 44], rich content of nodes are incorporated to obtain more discriminative representations. Other extensions include dynamic [25], heterogeneous [13, 43], document [23, 38], community preserving [41] and emerging [47] network embedding. In addition to that, many other deep learning based approaches are proposed to further enhance different learning tasks [8, 32, 40]. Our work differs from these approaches as we first study how to perform network embedding in a multi-layered network.

6 Conclusions

Multi-layered networks emerge to be a new network representation and naturally arises themselves in various domains. With the prevalence of the multi-layered networks in many real-world applications, there are a few attempts to study this complex network representation in order to better understand it and derive some actionable patterns upon it. Network embedding has shown its power in modeling traditional single-layered networks as the learned node embedding representation could advance many graph mining tasks. However, the study on how to obtain embedding representation on multi-layered networks is still an unsolved problem. In this paper, we study this important problem by proposing a novel network embedding framework MANE. To capture both within-layer node connections and cross-layer node dependencies, MANE employs the spectral embedding and the block model to joint model different network dependencies. An effective alternating optimization algorithm is also presented for the proposed MANE framework. We validate the superiority of MANE on real-world multi-layered networks.

Future work can be focused on studying the multi-layered network embedding problem in more complex settings, for example in dynamic multi-layered networks and multi-layered networks with rich node attributes. Another potential research direction is to study task-oriented multi-layered network embedding.

Acknowledgements

Jundong Li and Huan Liu are supported by NSF (IIS-1614576) and ONR (N00014-16-1-2257). Chen Chen and Hanghang Tong are supported by NSF (IIS-1651203, IIS-1715385) and DTRA (HDTRA1-16-0017).

References

[1] L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *WSDM*, 2011.

[2] F. Battiston, V. Nicosia, and V. Latora. Structural measures for multiplex networks. *Physical Review E*, 2014.

[3] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, and D. Pedreschi. Foundations of multidimensional network analysis. In *ASONAM*, 2011.

[4] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In *SNAM*. 2011.

[5] R. Bhatia. *Matrix Analysis*. 2013.

[6] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 2010.

[7] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, 2015.

[8] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *KDD*, 2015.

[9] C. Chen, J. He, N. Bliss, and H. Tong. On the connectivity of multi-layered networks: models, measures and optimal control. In *ICDM*, 2015.

[10] C. Chen, J. He, N. Bliss, and H. Tong. Towards optimal connectivity on multi-layered networks. *TKDE*, 2017.

[11] C. Chen, H. Tong, L. Xie, L. Ying, and Q. He. Fascinate: fast cross-layer dependency inference on multi-layered networks. In *KDD*, 2016.

[12] L. Chen, X. Xu, S. Lee, S. Duan, A. G. Tarditi, S. Chinthavali, and B. A. Prakash. Hotspots: Failure cascades on heterogeneous critical infrastructure networks. In *CIKM*, 2017.

[13] T. Chen and Y. Sun. Task-guided and path-augmented heterogeneous network embedding for author identification. In *WSDM*, 2017.

[14] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: scalable representation learning for heterogeneous networks. In *KDD*, 2017.

[15] S. Fortunato. Community detection in graphs. *Physics Reports*, 2010.

[16] J. Gao, S. V. Buldyrev, H. E. Stanley, and S. Havlin. Networks formed from interdependent networks. *Nature Physics*, 2012.

[17] A. Grover and J. Leskovec. Node2vec: scalable feature learning for networks. In *KDD*, 2016.

[18] L. S. Heath and A. A. Sioson. Multimodal networks: structure and operations. *IEEE/ACM TCBB*, 2009.

[19] X. Huang, J. Li, and X. Hu. Accelerated attributed network embedding. In *SDM*, 2017.

[20] X. Huang, J. Li, and X. Hu. Label informed attributed network embedding. In *WSDM*, 2017.

[21] L. Jian, J. Li, and H. Liu. Toward online node classification on streaming networks. *DMKD*, 2017.

[22] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of Complex Networks*, 2014.

[23] T. M. Le and H. W. Lauw. Probabilistic latent document network embedding. In *ICDM*, 2014.

[24] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

[25] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu. Attributed network embedding for learning in a dynamic environment. In *CIKM*, 2017.

[26] J. Li, C. Kewei, L. Wu, and H. Liu. Streaming link prediction on dynamic attributed networks. In *WSDM*, 2018.

[27] J. Li, L. Wu, O. R. Zaıane, and H. Liu. Toward personalized relational learning. In *SDM*, 2017.

[28] B. Long, Z. M. Zhang, and P. S. Yu. Co-clustering by block value decomposition. In *KDD*, 2005.

[29] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002.

[30] J. Ni, H. Tong, W. Fan, and X. Zhang. Inside the atoms: ranking on a network of networks. In *KDD*, 2014.

[31] J. Ni, H. Tong, W. Fan, and X. Zhang. Flexible and robust multi-network clustering. In *KDD*, 2015.

[32] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang. Tri-party deep network representation. In *IJCAI*, 2016.

[33] R. Parshani, S. V. Buldyrev, and S. Havlin. Interdependent networks: reducing the coupling strength leads to a change from a first to second order percolation transition. *Physical review letters*, 2010.

[34] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: online learning of social representations. In *KDD*, 2014.

[35] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Systems*, 2001.

[36] J. Shao, S. V. Buldyrev, S. Havlin, and H. E. Stanley. Cascade of failures in coupled network systems with multiple support-dependence relations. *Physical Review E*, 2011.

[37] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *KDD*, 2008.

[38] J. Tang, M. Qu, and Q. Mei. Pte: predictive text embedding through large-scale heterogeneous text networks. In *KDD*, 2015.

[39] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: large-scale information network embedding. In *WWW*, 2015.

[40] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *KDD*, 2016.

[41] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang. Community preserving network embedding. In *AAAI*, 2017.

[42] D. J. Watts and S. H. Strogatz. Collective dynamics of small-worldnetworks. *Nature*, 1998.

[43] L. Xu, X. Wei, J. Cao, and P. S. Yu. Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks. In *WSDM*, 2017.

[44] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang. Network representation learning with rich text information. In *IJCAI*, 2015.

[45] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM*, 2013.

[46] Y. Yao, H. Tong, G. Yan, F. Xu, X. Zhang, B. K. Szymanski, and J. Lu. Dual-regularized one-class collaborative filtering. In *CIKM*, 2014.

[47] J. Zhang, C. Xia, C. Zhang, L. Cui, Y. Fu, and P. S. Yu. Bl-mne: Emerging heterogeneous social network embedding through broad learning with aligned autoencoder. In *ICDM*, 2017.

[48] D. Zhou, J. He, K. S. Candan, and H. Davulcu. Muvir: Multi-view rare category detection. In *IJCAI*, 2015.

[49] Y. Zhou and J. He. A randomized approach for crowdsourcing in the presence of multiple views. In *ICDM*, 2017.